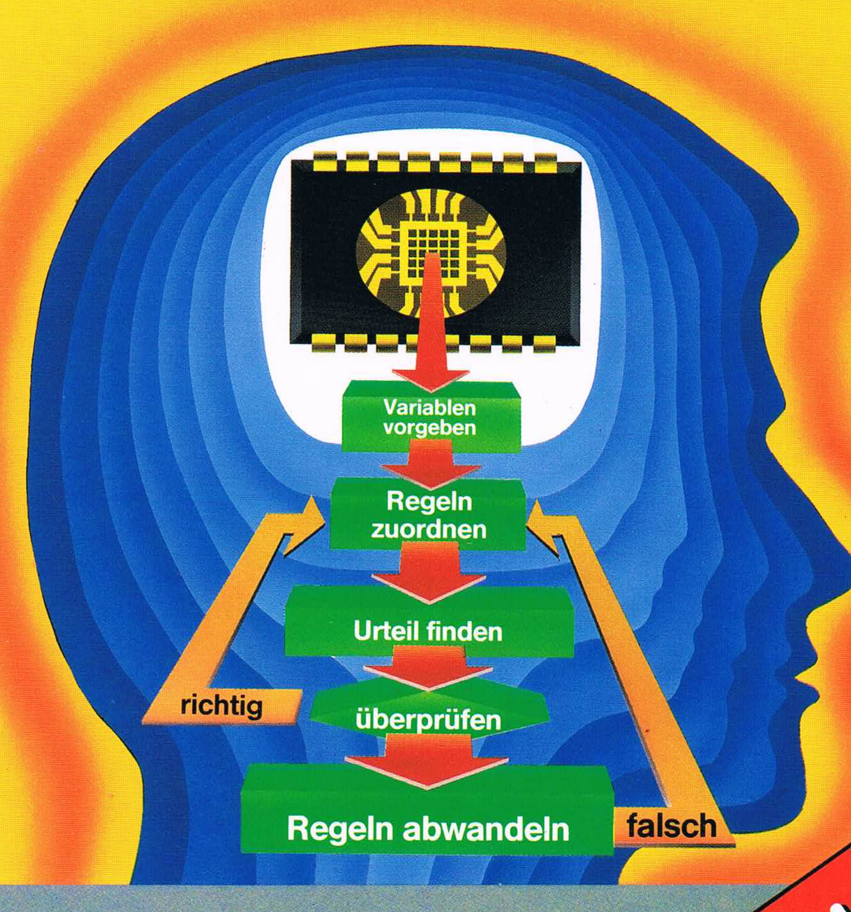
Computer verständlich

Karl-Heinz Koch

Intelligenz in BASIC

Für Schneider CPC 464/664/6128



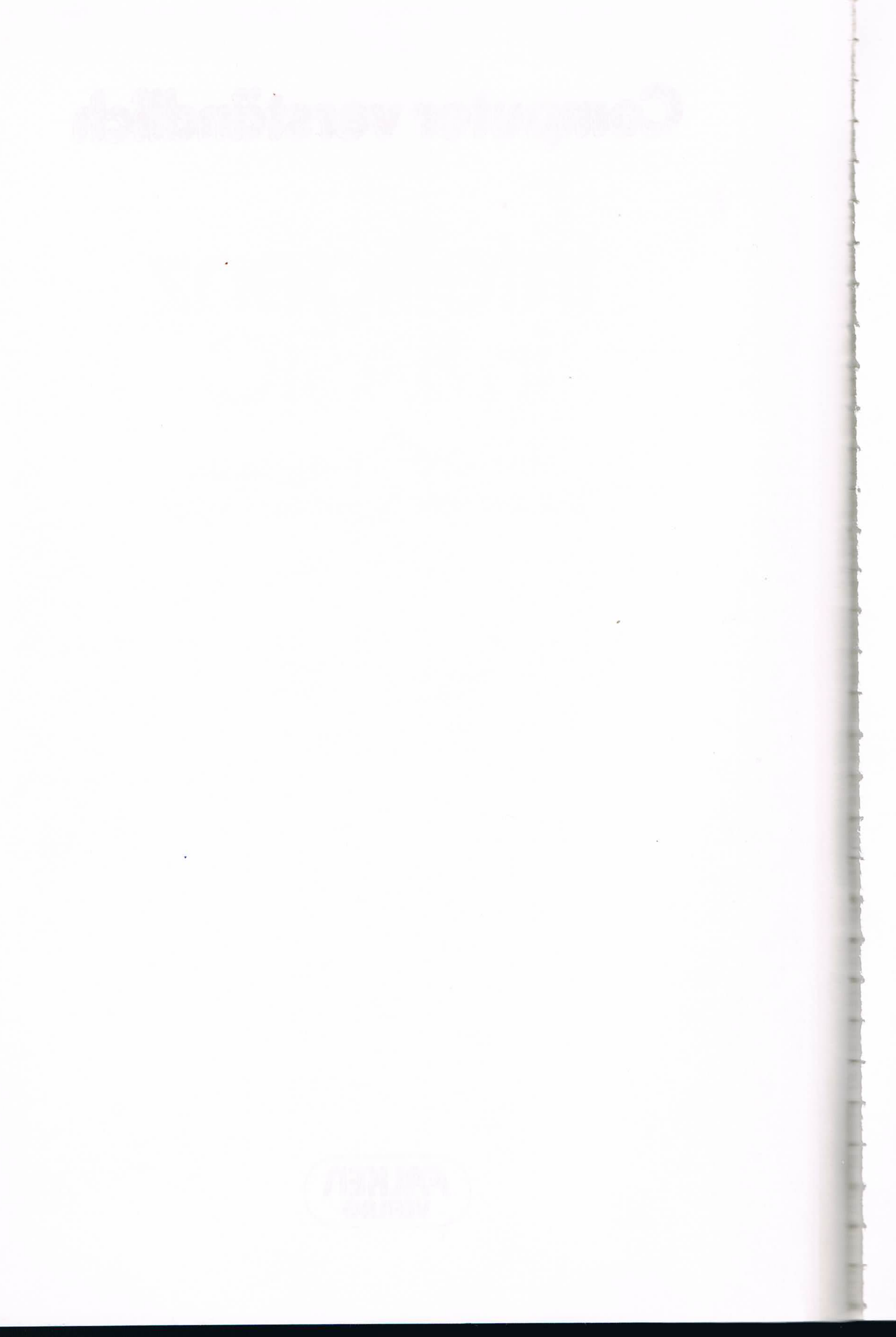
Mit Diskette 3"

FALKEN VERLAG



Computer verständlich





Karl-Heinz Koch

Intelligenz in BASIC

Vom ersten Programm bis zur intelligenten Datei



Weitere Titel aus der Reihe »Computer verständlich«:
»Drucker und Plotter« von Karl-Heinz Koch (Nr. 4315)
»Garantiert BASIC lernen mit dem C 128« von Alfred Görgens (Nr. 4321)
»Heimcomputer-Bastelkiste« von Gerhard A. Karl (Nr. 4309)
»Grundwissen Informationsverarbeitung« von Helmut Schiro (Nr. 4314)

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Koch, Karl-Heinz:
Intelligenz in BASIC: vom ersten Programm bis zur
intelligenten Datei / Karl-Heinz Koch.
— Niedernhausen/Ts.: Falken-Verlag, 1987.
(Computer verständlich)
ISBN 3-8068-4320-1

ISBN 3 8068 4320 1

Titelgestaltung: Kreativ-Design Gerd Aumann, Wiesbaden
Zeichnungen: Hans-Georg Stroh, Erftstadt-Kierdorf.

Satz: Dinges + Frick, Wiesbaden
Druck: Gebr. Achilles, Aachen

817 2635 4453 6271

Inhalt

Einleitung

Künstliche Intelligenz auf dem Heimcomputer? Und dann noch in BASIC?
Sicher, diese hochspezialisierte Anwendung verlangt nicht nur fundiertes Informatikwissen, sondern auch spezielle Programmiersprachen und entsprechend konstruierte Computer. In den Bereichen Soft- und Hardware liegen heute zwar verschiedene Entwicklungen vor, doch für den interessierten Laien sind sie noch unerreichbar.

Nur keine falsche Ehrfurcht!

Dennoch, einige grundlegende Denkansätze lassen sich auch mit einfacher Ausrüstung nachvollziehen. Wer bereits Erfahrungen mit BASIC gesammelt hat, kann sich die Programmbeispiele in diesem Buch schnell aneignen, einen Einblick in diese neue Entwicklungsrichtung der Computertechnik gewinnen und den ehrwürdigen Schauer vor dem fremden Gebiet verlieren.
Lernfähige Programme, ein erster Schritt zur »Künstlichen Intelligenz« (**KI**) oder »Artificial Intelligence« (**AI**), lassen sich auch in BASIC schreiben. Und nichts motiviert mehr, sich mit KI eingehender zu beschäftigen, als zu erleben, wie so ein lernendes Programm reagiert.
Eine Ratte sucht einen Weg durch ein Labyrinth, merkt sich Irrwege und kommt mit jedem neuen Versuch schneller zum Ausgang, bis sie schließlich den direkten Weg kennt.
Ein Brettspiel: Der Computer kennt nur die Regeln, aber mit jeder Partie lernt er hinzu, und schließlich ist er nicht mehr zu schlagen.
Ein Programm stellt Fragen und lernt dabei, beliebige Begriffe zu unterscheiden.
Diese Computererlebnisse machen nicht nur Spaß, sie nehmen auch die Scheu vor dem geheimnisvollen Neuland. Künstliche Intelligenz oder Artificial Intelligence heißt in jüngster Zeit die Zauberformel, mit der die Magier der Hochtechnologie Öffentlichkeit und Geldgeber in Politik und Wirtschaft bannen. Mit ihrem KI-Projekt der »fünften Computergeneration« haben die Japaner den Kampf um die Spitzenposition in der Computertechnik eingeläutet. Doch ganz so neu ist dieser Forschungsbereich gar nicht.

KI und die Entwicklung der Computer-Technologie

Die Spitzentechnologien werden mit rasantem Nachdruck vorangetrieben, denn politische und wirtschaftliche Interessen stehen dahinter. Die westlichen Staaten sehen ihre Überlegenheit auf diesem Gebiet — nicht

nur wirtschaftlich. Die KI-Forschung wird zum großen Teil aus den Haushalten der Verteidigungsministerien finanziert. Der großartige Begriff »Künstliche Intelligenz« macht die angestrebte Überlegenheit sprachlich faßbar.
Die wirtschaftlich verwertbaren Ergebnisse dieser Anstrengungen haben einen umsatzstarken Industriezweig hervorgebracht. Großes Geld ist zu verdienen, doch nicht minder groß ist die Konkurrenz. Da Computer, Roboter und die Programme, die sie bewegen, zu den Investitionsgütern gehören, muß den Entscheidungsträgern der Wirtschaft im besonderen und der Öffentlichkeit im allgemeinen der Eindruck großer Leistungsfähigkeit vermittelt werden. Auch hierzu taugt das Wortpaar »Künstliche Intelligenz« trefflich.

Viel Lärm um nichts...

Daß dieser Begriff jetzt so hochgespielt wird, hat allerdings nicht nur propagandistische Hintergründe. Die KI-Forschung geht bis in die 50er Jahre zurück, praktische Erfolge hatte sie bislang nur wenige vorzuweisen, und diese wenigen Anwendungen eigneten sich nicht für eine breitere wirtschaftliche Verwertung. In den letzten Jahren hat sich die Datenverarbeitung stark weiterentwickelt. Die Computer sind kleiner und leistungsfähiger geworden. Die Computerisierung der Gesellschaft ist recht weit vorangetrieben. Da scheint es notwendig, schon auf die nächste Ausbaustufe zu zeigen, um die Entwicklung weiter in Gang zu halten. Die Parole heißt: von der Datenverarbeitung zur Wissensverarbeitung. Werkzeuge dafür sollen die **Expertensysteme** sein, eine Anwendung von »Künstlicher Intelligenz«, die im Schatten dieses Wortes noch bescheiden wirkt.

...oder steckt doch etwas dahinter?

Die weltbewegenden Interessen hinter dieser Entwicklung haben zwischen Anhängern und Kritikern einen heftigen Streit darüber ausgelöst, ob es überhaupt möglich sei, Intelligenz in Maschinen zu reproduzieren.
Einen Beitrag zur KI-Diskussion kann und will dieses Buch nicht leisten. Der erste Teil gibt lediglich einen orientierenden Einblick in Entwicklung, Hintergründe, Denkansätze und widersprechende Auffassungen. Damit soll nur eine theoretische Einstimmung geschaffen werden, damit die praktischen Programmbeispiele in dem entsprechenden Sachzusammenhang gesehen werden können. Wer sich eingehender mit der theoretischen Diskussion auseinandersetzen will, findet im Literaturverzeichnis eine Auswahl lesenswerter Bücher.

Grau ist alle Theorie: Programme mit Farbe . . .

Die Programme in diesem Buch sind auf einem Schneider-CPC-6128-Heimcomputer geschrieben. Dabei wurden die reichen Farbmöglichkeiten dieses Gerätes berücksichtigt. Wenn Sie mit einem Monochrommonitor arbeiten, kann es hier und da notwendig sein, die Farbbefehle zu ändern oder einfach herauszunehmen.
Obwohl es sich auch bei den größeren Programmen um bescheidene Demonstrationen künstlicher Intelligenz handelt, sind die Listings durchweg recht umfangreich. Der Abdruck der Listings erfolgt in einer Breite von 40 Zeichen pro Zeile, also wie die Darstellung auf dem Bildschirm. Trotzdem kann es sehr ermüdend sein, diese langen Programme abzutippen. Diesem Buch ist eine Diskette beigelegt, auf der nicht nur alle Programme enthalten sind, sondern zusätzlich die Beispieldatei MOZART zu dem Expertensystem DENKNETZ, die sich natürlich nicht in einem Buch abdrucken läßt. Die Diskette ist nicht geschützt. Sie können die Programme also auflisten, kopieren und abändern. Ihrem Forscherdrang sind keine Hindernisse in den Weg gelegt.
Leider bietet der Schneider CPC keinen generellen Befehl, um alle ergangenen Farbbefehle in den Grundzustand zurückzusetzen. Dies kann nur erreicht werden, indem man die Tasten <SHIFT> und <CONTROL> gedrückt hält und dann die Taste <ESC> betätigt. Dadurch wird ein Warmstart des Rechners ausgelöst. Die Programme auf der Diskette enthalten alle die Zeile 5, welche die Grundwerte der Farben für Window #0 und den Bildschirmrand definiert. Wenn Sie nacheinander verschiedene Programme in den Rechner laden, besteht trotzdem die Möglichkeit, daß sich noch Farbinformationen für andere Windows im Speicher befinden, die das neue Programm beeinflussen. Sollte ein solcher Fall auftreten, setzen Sie den Computer bitte wie oben beschrieben zurück, und laden Sie das gewünschte Programm neu.

. . . und ohne

Soviel zur Technik, die ja nur ein wirkungsvolles Hilfsmittel für die Entfaltung unseres Bewußtseins sein sollte. Bei der Beschäftigung mit künstlicher Intelligenz wird das deutlich, denn die scheint ohne eine Auseinandersetzung mit unserer menschlichen Intelligenz nicht besonders sinnvoll. Deshalb wünsche ich Ihnen auf der Expedition in das unbekannte Land Ihrer eigenen Gehirnwindungen recht viel Spaß und viele überraschende Entdeckungen. Vielleicht finden Sie sogar eine Schatzinsel. Chip Ahoi!

Intelligenz künstlich?

Computer mit Gefühlen?

Können Sie sich vorstellen, daß Ihr Computer in Sie verliebt ist?
Sie betätigen den Netzschalter, laden das Betriebssystem, und auf dem Monitor erscheint rührend unbeholfen in hochauflösender Grafik ein Herz, von einem Pfeil durchbohrt. Darüber leuchtet in blinkenden Buchstaben der Text: »Ich liebe Dich.«
Sicher würden Sie diesen Vorfall als nicht besonders gelungenen Gag des Informatikers einstufen, der das Programm entworfen hat, das Sie gerade geladen haben. Sie würden sicher auch nicht im entferntesten daran glauben, daß Frühlingsgefühle die Schaltkreise Ihres Micros durchwehen.
Jeder vernünftige Mensch wird Computern die Fähigkeit zu Gefühlen abstreiten und die Sicherheit seines Urteils aus dem gegenwärtigen Stand der technischen Entwicklung herleiten. Aber warum scheiden sich die Geister, wenn die Frage nur ein wenig verändert wird: Werden Computer irgendwann einmal Gefühle haben können?

Im Film ist manches möglich, in der Realität wird es zumindest noch ein Weilchen dauern . . .

In dem Kinofilm »2001: Odyssee im Weltraum« ist der Bordcomputer so programmiert, daß er gefühlvoll wie ein menschliches Wesen reagiert. Den Astronauten soll damit auf ihrer langen, unwirtlichen Reise ein Gefühl heimischer Geborgenheit vermittelt werden. Das Ergebnis ist verheerend, weil die Maschine auch in einer Krisensituation gefühlvoll reagiert: Sie ist unfähig, einen Fehler zuzugeben. Trotzdem fällt die Schuld nicht der Maschine zu, sondern ihren Programmierern und dem Konzern, dessen Interessen sich darin spiegeln. — Es kann doch nicht angehen, daß eine Maschine Empfindungen hat, noch dazu menschliche, oder?

Bei der KI geht es nicht um Nachahmung der menschlichen Intelligenz

Nun, die Informationsverarbeitung im menschlichen Gehirn unterliegt so vielschichtigen Vorgängen, daß heute bei weitem noch nicht daran zu denken ist, einen Computer zu bauen, der nach dem gleichen Prinzip arbeiten kann. Das ist auch gar nicht die Frage, mit der uns das Konzept der »Künstlichen Intelligenz« (KI) konfrontiert, so lange die Begründung nur in der augenblicklichen Kapazität der Geräte (Hardware) und bisherigen Leistungsfähigkeit der Programme (Software) gesehen wird.

Die eigentliche Frage, von der die Gemüter ernsthafter Wissenschaftler auf der ganzen Welt seit einigen Jahren so sehr erhitzt werden, lautet, ob es im Prinzip möglich sei, eine Maschine zu konstruieren, deren Datenverarbeitungsergebnisse von menschlichen Denkleistungen nicht zu unterscheiden sind oder diese gar übertreffen. Wenn man es aber für möglich hält, irgendwann einmal denkende Maschinen zu bauen, muß man es dann nicht auch für möglich erachten, daß diese Maschinen auf einer weiteren Entwicklungsstufe Gefühle entwickeln können?

Maschinenstürmer und Phantasten

Der Mensch als Schöpfer — ein uralter Traum

Künstliche Intelligenz zu schaffen ist ein uralter Menschheitstraum, ähnlich dem Traum vom Fliegen. Er ist sicher so alt, wie Menschen über ihr Denken nachdenken. Das Vermögen, kraft seines Geistes aus toter Materie Leben zu erschaffen, würde den Menschen um eine weitere Stufe dem Göttlichen näherbringen.
In der griechischen Mythologie begegnet uns dieses Thema bereits in der Figur des Pygmalion. Ein Künstler schafft eine vollendete weibliche Statue aus Elfenbein, in die er sich darauf so sehr verliebt, daß er zu den Göttern fleht, sie mögen sie zum Leben erwecken. Aphrodite, Göttin der sinnlichen Liebe und der Schönheit, erhört schließlich die Bitte, und der Künstler kann sich mit dem Objekt seiner Kunst vermählen.
Diese von Ovid erzählte Geschichte rührt so sehr an den Kern menschlicher Sehnsüchte, daß sie über die Jahrtausende hinweg immer wieder von Komponisten, Dichtern und Malern aufgegriffen und neu bearbeitet wurde.
Während die Träume die gleichen blieben, wandelten sich die Methoden mit dem technisch-wissenschaftlichen Fortschritt. Trotzdem gab und gibt es zwei grundsätzlich verschiedene Ansätze bei dem Unternehmen der künstlichen Neuschaffung des Menschen durch sich selbst. Die Anhänger dieser beiden Richtungen könnte man als die »Biologen« und die »Mechanisten« bezeichnen. Welche Gruppierung vorübergehend größere Bedeutung erlangte, hing jeweils von der Entwicklung wissenschaftlicher Erkenntnisse ab.

»Biologen« kontra »Mechanisten«

Bereits im 11. Jahrhundert behauptete Papst Sylvester III., im Besitz einer Maschine zu sein, die Fragen mit »ja« oder »nein« beantworten könne. Arabische Astrologen sollen im Mittelalter eine Maschine gebaut haben, von der man erzählte, daß sie sogar philosophischen Diskussionen folgen könne. Typisch ist, daß Maschinen dieser Art Namen bekamen, die arabische Denkmaschine hieß »Zairja«, und daß der Ursprung dieser Wunderapparate gern in exotische Länder verlegt wurde.
Menschlein, aus einer Alraune gezüchtet, tummelten sich lange in den Köpfen der Träumer. Mit dem Beginn medizinischer Forschung verbreitete sich jedoch schnell der Glaube, menschliches, biologisches Leben sei künstlich zu erschaffen. Selbst Paracelsus, der am Ende des 15. Jahrhunderts einen Lehrstuhl für Physik und Medizin in Basel hatte, behauptete noch, ein Rezept für die Herstellung eines Humunculus, eines künstlichen Menschen, zu besitzen. Gebraut wurde gewöhnlich mit Sperma, Blut und anderer Biomasse. Und damals wie heute war die Phiole oder die Retorte dabei.

Die ersten »Roboter«

Größere Erfolge hatten jedoch bald die Mechanisten vorzuweisen. Mit der Entwicklung der Feinmechanik wurde der Bau von Maschinen möglich, die schlichte Gemüter, und dazu gehörten nicht selten auch kirchliche wie weltliche Fürsten, glauben machten, Intelligenz sei mechanisch zu erzeugen. Immerhin gelang die Konstruktion von Puppen, die, von einem Uhrwerk angetrieben, einen echten Federkiel in ein echtes Glas Tinte tauchten und auf echtem Pergament einen Namen schrieben, eine Zeichnung fertigten oder die Noten einer Melodie niederschrieben. Das fasziniert noch heute jeden Museumsbesucher.
In der jüdischen Mystik begegnet uns die künstliche Intelligenz als »formlose Masse«, so die Übersetzung des Wortes »Golem«, ein Wesen, aus Lehm oder Ton geknetet und von Weisen und Magiern mittels Buchstabenmagie zum Leben erweckt. Das erinnert an das erste Buch Mose, die Erschaffung des Menschen aus Erde und göttlichem Odem, und es zeigt, wie sehr die Schaffung künstlicher Intelligenz mit der Göttlichkeitssehnsucht des Menschen verbunden ist. Der Golem ist allerdings nur von beachtlicher Größe und Kraft, zum Sprechen reicht seine Intelligenz nicht aus.
In der zweiten Hälfte des 16. Jahrhunderts will der Hohe Rabbi Löwe Juda Ben Bezahel aus dem Lehm des Moldauufers noch einen solchen Golem in die Welt gesetzt haben, der danach nicht nur durch Prag stakste, sondern darüber hinaus während der nächsten vierhun-

Golem und Frankenstein

dert Jahre Dichter wie Jacob Grimm, Achim von Arnim, E. T. A. Hoffmann, Annette von Droste-Hülshoff oder Gustav Meyrink beschäftigte.

Erst als die Schriftstellerin Mary W. Shelley ihren wahnsinnig-genialen Professor »Frankenstein« aus Leichenteilen zusammennähen ließ, konnte der tönerne Golem in den wohlverdienten Ruhestand treten.

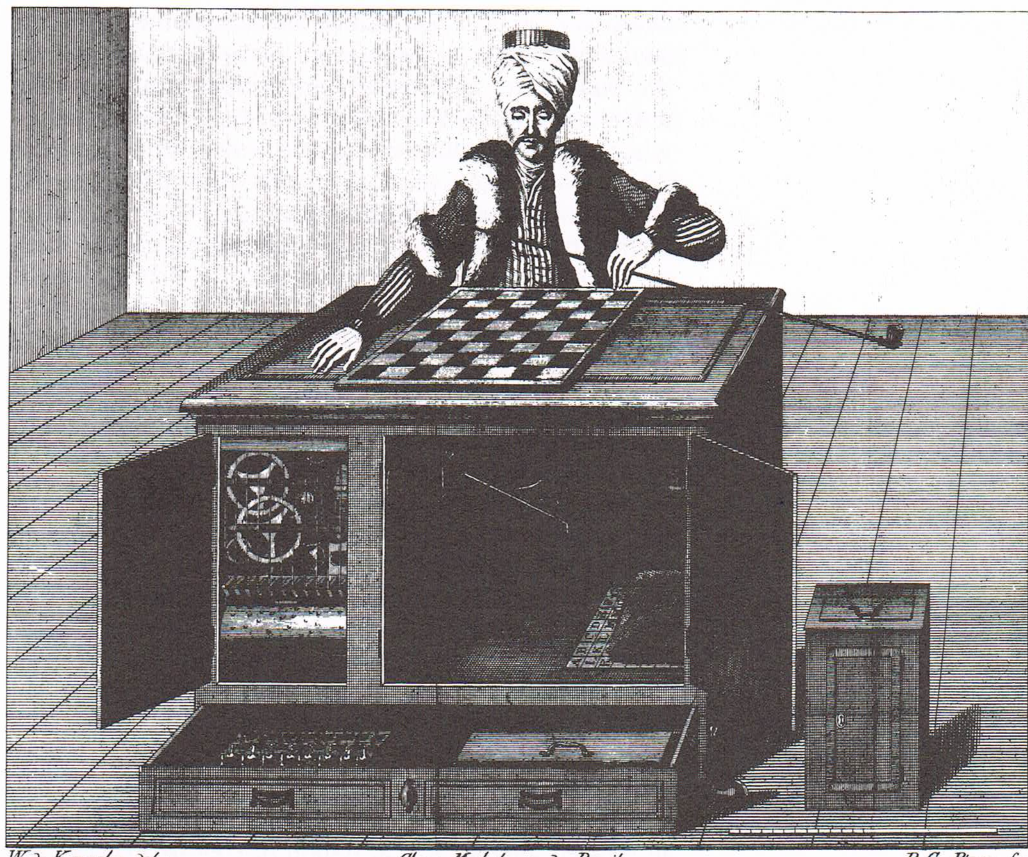
Die soziale Symbolik des Frankensteinschen Monsters ist von der schnellen Trivialisierung der Figur überrollt worden. Fortschritte in Medizin und Chirurgie hatten in der Gesellschaft jener Zeit Ängste anwachsen lassen, die sich auf das mißratene Kunstwerk projizieren ließen. Vom erbaulichen, mechanischen Spielzeug hatte sich der künstliche Mensch zum schwer kontrollierbaren und damit gefährlichen Produkt einer Wissenschaft entwickelt, die für den Laien so unverständlich wurde, daß er sie nur im Bereich des Magischen anzusiedeln vermochte.

Doch während die Möglichkeiten der Wissenschaftler, verändernd in das Leben einzugreifen, dem gesunden Menschenverstand immer unvorstellbarer und wahnsinniger erschienen, ging gleichzeitig die Glaubwürdigkeit verloren. Zu viele Scharlatane mischten in diesem blendenden Spiel mit.

So »baut man Türken«: Die Herkunft eines Sprichwortes

1770 baute der Hofrat W. v. Kempelen seinen »Türken« und präsentierte ihn am Hofe Maria Theresias. Auf einer Holzkiste saß eine fast lebensgroße Figur vor einem Schachbrett mit frei beweglichen Figuren. Vor Beginn der Demonstration öffnete der Hofrat eine Tür an der Vorderseite der Kiste und gab den Einblick frei auf ein verwirrendes Innenleben aus Walzen, Zahnrädern und Treibriemen. Verschiedene Schachmeister, alle von kleinem Wuchs, fanden im Lauf der Jahre in dieser Kiste Platz, um durch einen sinnreichen Mechanismus die Partie im Verborgenen zu verfolgen und den linken Arm des Türken so zu bewegen, daß er die Züge ausführen konnte, die sie sich ausdachten. Fast zwanzig Jahre lang konnte das Geheimnis bewahrt werden.

Als mit dem Wirtschaftswunder auch Computer breitere gesellschaftliche Bedeutung erhielten, bürgerte sich schnell das Wort »Elektronengehirn« ein. Ein neuer Typ Golem war geboren. Schon wenige Jahre später lag auf jedem Schreibtisch ein Taschenrechner, und das Wort kam wieder aus der Mode. Aber die ersten Maschinen, die, wie schlecht auch immer, Schach spielen konnten, sorgten für anhaltendes Aufsehen und vor allem für gute Umsätze.



Schon immer faszinierte die Vorstellung, eine denkende Maschine zu bauen oder einen künstlichen Menschen zu schaffen. Der »Türke« des Barons von Kempelen erzeugte allerdings nur eine Illusion. Der Schachspieler wurde von einem Menschen bewegt, der im Inneren der Apparatur verborgen war.

Soziale und psychologische Aspekte

Technische Neuerungen dienen dazu, dem Menschen die Mühe zu erleichtern, sich die Erde untertan zu machen, aber sie verdrängen auch immer Arbeiter von ihren gewohnten Plätzen. Das führt zu sozialen Unruhen. So war es, als der mechanische Webstuhl eingeführt wurde, und so ist es heute, wo Computer und Roboter die Produktionsabläufe umgestalten. Doch während bislang die »Künstliche Intelligenz« im Reich des Mystischen nur Ausdruck geheimer Ängste und Hoffnungen war, scheint der Golem nun aus dem Zwielicht der Träume in die alltägliche Realität herauszutreten.

Die psycho-sozialen Konflikte, die sich hier abzeichnen, sind Stoff für ein anderes Buch. Sie zeigen aber, warum die Diskussion um künstliche Intelligenz so heftig — und meist auch so sehr von Gefühlen bestimmt — geführt wird. Maschinenstürmer haben die technische Entwicklung nie aufhalten können. Immer wieder hat sich gezeigt, daß Angst vor Veränderung und Unkenntnis der Sache die Betroffenen mehr bewegte, als die Technik wirklich das Leben bedrohte.
Computer, Roboter und die künstliche Intelligenz, die sie bewegt oder einmal bewegen soll, sind nicht das Problem, sondern die gesellschaftlichen Veränderungen, die sie mit sich bringen, und die gerechte Verteilung der Vor- und Nachteile, die daraus erwachsen. Der Herausforderung der Hochtechnologie ist nicht dadurch zu begegnen, daß wir sie wie einen Golem fürchten, sondern durch Lernen und Fortentwickeln unserer eigenen Fähigkeiten, Herr der Dinge zu bleiben und zu unserem Vorteil zu nutzen, was menschlicher Erfindergeist geschaffen hat.

Expertenstreit

In einem Bunker, tief im Bauch der Erde, arbeitet ein Team von Wissenschaftlern an einem geheimen Regierungsprojekt, einem Supercomputer, der so viel Wissen speichern und es so schnell verarbeiten kann wie keine Maschine vorher. Nach seiner Fertigstellung erhofft man sich Antwort auf Fragen, die über die schlichte Datenverarbeitung hinausgehen und den Erkenntnishorizont erweitern sollen. Die Funktionen der Maschine sind der Arbeitsweise menschlicher Intelligenz nachgebildet. Als der elektronische Riese nach langen Jahren harter Arbeit endlich in Betrieb genommen werden kann, gibt der Teamleiter als erstes eine Frage ein, die von den Experten als die wichtigste ausgewählt worden war: »Gibt es einen Gott?« Ohne auch nur eine merkliche Bearbeitungszeit zu benötigen, antwortet das Elektronengehirn: »Von jetzt an schon.«

Gibt es bald den eingebildeten Computer?

Wir haben uns daran gewöhnt, daß Computer in ihren Tätigkeitsbereichen in beängstigendem Maße schneller und präziser arbeiten, als Menschen es vermögen. Deshalb verbindet sich mit der Vorstellung von intelligenten Computern die Angst vor einer höheren Intelligenz als der menschlichen und der endgültigen Überlegenheit der Maschine.

Die Japaner erregten beträchtliches Aufsehen, als sie die fünfte Generation der Computer ankündigten. Bis 1990 wollen sie intelligente Maschinen entwickeln. Forscher, die weniger auf spektakuläre Medienwirkung schielen, sind sich allerdings noch nicht ganz einig darüber, was überhaupt eine intelligente Maschine ist. Man kann sich vorstellen, alle heute bekannten Wissensinhalte in einer Datenbank zu erfassen, um sie von einer Maschine durchforsten und neu kombinieren zu lassen. Selbst wenn man davon absieht, daß es praktisch kaum möglich ist, diese gewaltige Datenmenge entsprechend aufzuarbeiten und einzugeben, ist doch noch nicht erwiesen, daß eine Maschine selbständig neue Erkenntnisse gewinnen kann.

Es besteht keine Einigkeit darüber, was Intelligenz überhaupt ist und wie intelligente Prozesse im menschlichen Gehirn ablaufen. Deshalb ist noch offen, ob und in welchem Maße sich Denken außerhalb des Körpers, also in einem Apparat, reproduzieren läßt.

Das Vorbild des menschlichen Gehirns

Wie rätselhaft die Denkvorgänge heute noch immer für uns sind, zeigt die Beschreibung eines Gehirnforschers. Er vergleicht das menschliche Gehirn mit einem gigantischen Büro mit 15 Milliarden Angestellten, die alle gleichzeitig miteinander telefonieren. Wie aus der Summe unserer Sinneseindrücke und der Vielzahl chemischer und elektrischer Aktivitäten im Gehirn Gedanken entstehen, wissen wir nicht, und nach welchen Prinzipien diese Gedanken intelligent werden, schon gar nicht.

Ein »kleiner« Trick

Wer nicht exakt beschreiben kann, was als Intelligenz verstanden werden soll, tut sich folglich auch schwer zu definieren, was künstliche Intelligenz ist, oder wann eine Maschine intelligent arbeitet. Der Computerwissenschaftler Alan Turing hat sich einen Trick zurechtgelegt, dieses Dilemma zu umgehen. Ein nach ihm benannter Test ist so aufgebaut, daß eine Versuchsperson an einem Terminal sitzt und ihren Dialogbeitrag über die Tastatur eingibt. Der zweite Dialogteil wird von einem Rechner auf dem Monitor ausgegeben. Wenn die Versuchsperson nicht entscheiden kann, ob ihr Kommunikationspartner ein Mensch oder eine Maschine ist, dann soll dem Computer Intelligenz zugesprochen werden.

So überzeugend diese Konstruktion auf den ersten Blick erscheinen mag, entpuppt sie sich doch als problematisch, wenn man versucht, sie anzuwenden. Das Urteil der Testperson ist abhängig von ihrer eigenen Intelligenz, also von keiner exakt bestimmbaren Größe, und vom Gesprächsthema. Wie leicht es ist, einem ahnungs-

losen Menschen intelligentes Verhalten nach dem Modell der Turing-Maschine vorzutäuschen, hat der dem hochtechnologischen Fortschritt skeptisch gegenüberstehende Joseph Weizenbaum vom Massachusetts Institute of Technology (MIT) 1964 mit seinem Programm ELIZA demonstriert. ELIZA reagiert ähnlich wie ein Psychiater beim Erstgespräch mit einem Patienten. Das Programm untersucht die Eingaben der Versuchsperson nach grammatikalischen Strukturen und Stichwörtern und bildet daraus nach vorgegebenen Schemata Antworten und Nachfragen. Solange sich der Dialog im Bereich unverbindlicher Aussagen ohne Bezug auf Sachwissen bewegt, vermittelt dieses Programm durchaus den Eindruck eines aufmerksamen und anteilnehmenden Zuhörers. Tatsächlich manipuliert das Programm aber nur die Eingaben und vermischt sie mit einer begrenzten Anzahl vorgegebener Muster. Mit der Arbeitsweise eines solchen Dialogprogramms können Sie sich in einem späteren Kapitel vertraut machen. DEPRESS.DIA arbeitet nach ähnlichen Prinzipien wie ELIZA, erwartet aber speziell Eingaben von Benutzern, die unter dem Einfluß von Depressionen und Lebensmüdigkeit stehen.

Schein und Sein

Scheinbar intelligent reagierende Programme dieser Art veranschaulichen den Konflikt in der KI-Diskussion. Eine Maschine ist lediglich ein totes Medium, das elektronische Impulse verarbeitet. Und dennoch vermag sie den Anschein intelligenten Handelns zu erwecken. Stellt man sich das menschliche Gehirn als eine biologische Maschine vor, die im Prinzip nicht anders funktioniert, wird es schwierig, wenn nicht unmöglich, zwischen der toten Maschine und dem lebenden Menschen zu unterscheiden. Setzt man voraus, daß sich menschliche Intelligenz nicht endgültig »mechanisch« beschreiben läßt, dann stellt sich die Frage, wodurch sie sich unterscheidet.

Der Streit um die KI und sein Hintergrund:
Materialismus oder mehr?

In der Regel vertreten die KI-Wissenschaftler die materialistisch-technische Auffassung, daß sich mit entsprechendem Wissen alles nachbauen lasse, also auch Intelligenz künstlich erzeugbar sei. Die Skeptiker und Kritiker beziehen meist eine philosophisch-spirituelle Position, die das Besondere des menschlichen Wesens hervorhebt, das sich, materialistisch argumentiert, schon in der Komplexität der Vorgänge zeige. So ließen sich theoretisch auch die Bewegungen des Meeres vorherbestimmen, wenn man nur die Bewegungen aller Wassermoleküle entsprechend addiere und die auf sie einwirkenden Kräfte fortlaufend berücksichtige. Die zur Berechnung notwendigen Formeln seien bekannt.

Trotzdem wäre ein solcher Versuch in der Praxis zum Scheitern verurteilt; und ähnlich verhielte es sich mit dem Versuch, Intelligenz nachzubilden.
John McCarthy von der Stanford University, einer der Pioniere der KI, resümiert etwas abgeklärt, die Verwirklichung künstlicher Intelligenz stünde nicht unmittelbar vor der Tür. Heute lasse sich noch nicht sagen, wie lange dies noch dauern werde. Er glaube jedoch, daß Intelligenz verstanden und in Maschinen neu geschaffen werden könne. Die Schätzungen von Experten, wieviel Entwicklungszeit noch benötigt wird, bis Maschinen wirklich intelligent funktionieren, reichen denn auch von 10 bis 1000 Jahre.
Zu den Skeptikern gehört der Philosoph Hubert Dreyfus von der Berkeley Universität, der das Grundkonzept der KI für verfehlt hält. Intelligenz sei nicht in starren Regeln zu erfassen. Für menschliches Verhalten wesentliche Faktoren wie Intuition oder Weisheit würden bei einem solchen Verfahren nicht berücksichtigt. Andere Kritiker verweisen ähnlich auf das sogenannte graue Wissen: die Erfahrung, den »Riecher« oder siebenten Sinn, den Instinkt.

Was heute machbar ist — und was nicht

Tatsächlich zeigen sich die Probleme in der KI-Forschung immer wieder dort, wo man sie anfangs überhaupt nicht erwartet hat. So ist es relativ leicht, und hier sind inzwischen auch erste Erfolge vorzuweisen, Spezialwissen in einem Expertensystem zu erfassen und dieses als Denkwerkzeug zur Verfügung zu stellen. Hingegen sind Vorgänge, die den meisten Menschen selbstverständlich erscheinen, nur mit großem Aufwand oder überhaupt nicht zu fassen. Ihnen liegt Alltagswissen zugrunde.
So ist bislang zum Beispiel auch die mechanische Übersetzung von Texten in eine Fremdsprache gescheitert. Gerade auf der Höhe des kalten Krieges waren amerikanische Regierungsbehörden an diesem Projekt, das zu den ersten der KI-Forschung zählt, besonders interessiert. Es machte auch keine Schwierigkeiten, die Inhalte von Wörterbüchern und Grammatiken zu erfassen und von Programmen verarbeiten zu lassen. Doch das sinnrichtige Übersetzen von Texten setzt das Verständnis ihres Inhalts voraus. Jeder Leser wird die Bedeutung der folgenden beiden Sätze erkennen: »Hans fuhr in seinem Wagen. Er machte 150 Sachen.« Wie aber soll ein Programm unterscheiden, ob Hans oder der Wagen »150 Sachen« macht? Ein zweites Textbeispiel soll das Problem verdeutlichen: »Hans fuhr in seinem Wagen. Er machte eine Menge Kohle.« Den unterschiedlichen Bezug des Pronomens »er« zu erkennen, ist zum Beispiel

bei der Übertragung ins Englische notwendig. Bezieht sich das »er« auf »Hans«, muß es mit »he«, unter Bezug auf das »Auto« aber mit »it« übersetzt werden.
Menschliche, intelligente Leistungen basieren offensichtlich zu einem großen Teil auf Alltagswissen und Allgemeinbildung. Dabei handelt es sich um eine diffuse Versammlung von Inhalten, die mit den Mitteln der Datenverarbeitung nur sehr schwer in den Griff zu bekommen ist. Gerade die als intelligent einzustufenden Vorgänge scheinen sich immer auf ein weit verzweigtes Netz von Hintergrundinformationen zu beziehen, ein Wissen, das uns wie selbstverständlich zur Verfügung steht, weil wir es seit unserer Geburt in unserem Gehirn speichern und verarbeiten, das aber einer Maschine Punkt für Punkt eingegeben werden muß. Der Einsatz von Computern ist überall dort sinnvoll, wo komplizierte Kalkulationen und Vergleiche durchzuführen und Rückgriffe auf große Datenmengen notwendig sind. Alltägliche, simple Vorgänge, die der Mensch schon in seiner Kindheit so verinnerlicht, daß er »keinen Gedanken mehr daran verschwenden muß«, stellen sich hingegen als schier unüberwindlicher Wust von Einzelschritten und Überlegungen dar, die sich bis jetzt noch der elektronischen Nachahmung entziehen. Als Alternative dazu, einer Maschine das komplette Alltagswissen eines Menschen manuell einzugeben, scheinen Systeme, die selbst lernen und Erfahrungen machen, eine erfolgversprechende Entwicklungsrichtung zu markieren.

Was für den Menschen selbstverständlich ist, fällt dem Computer schwer

Die Aufgabenstellung der computergesteuerten Sprachübersetzung wurde in den späten 50er Jahren in Angriff genommen und mit 20 Millionen Dollar staatlich gefördert. Bis heute ist noch kein nennenswerter Erfolg auf diesem Gebiet erzielt worden. Für KI-Projekte werden trotzdem immer noch gewaltige Summen zur Verfügung gestellt, weil es den Wissenschaftlern gelingt, die Politiker von der Relevanz dieser Forschungen zu überzeugen. Fast alle großen KI-Projekte werden von einem Verteidigungsministerium finanziert.
Zum SDI-Projekt merkt Hubert Dreyfus an, die Beteiligten gingen davon aus, daß die heutigen Waffensysteme zu schnell für die menschliche Reaktionsfähigkeit seien. So genüge im Ernstfall die zur Verfügung stehende Zeit nicht, um zu entscheiden, welche Rakete gezündet oder welcher Satellit abgeschossen werden soll. Es sei der Glaube an die KI-Forschung, der die Verantwortlichen davon abhalte, die wirklichen Gefahren zu beseitigen.

Was'n Intelligenz?

Möglicherweise werden spätere Generationen einmal feststellen, daß die größte Bedeutung der KI-Forschung, zumindest in der Anfangsphase, in der wir uns befinden, darin bestand, die Erforschung der Intelligenz auf eine neue Basis gestellt zu haben. Schließlich kann man nur das künstlich erschaffen, was man genau kennt, beziehungsweise man kann etwas nur in dem Maße nachbilden, wie man Kenntnis davon hat.

Bislang haben sich selbst die Psychologen, in deren Fachgebiet das Phänomen Intelligenz zuerst fällt, um eine sachgerechte Definition gedrückt. Letztlich haben sie sich auf das zurückgezogen, was E. G. Borings in folgende ironisierende Formel gekleidet hat: Intelligenz ist das, was mit einem Intelligenztest gemessen wird. Das bedeutet aber, daß Intelligenz jeweils etwas anderes ist, wenn man einen anderen Test zugrunde legt.

Die Antwort der Psychologen: »Nichts genaues weiß man nicht«

Eine präzise Durchleuchtung des Prozesses, wie Intelligenz arbeitet, stand bislang auch nicht im Zentrum der Bemühungen. Es gab ein viel größeres Interesse daran, Intelligenz zu messen, um aus den Ergebnissen eine Bewertung der verschiedenen Menschen abzuleiten. Auf ähnliche Weise soll zum Beispiel aus Schulnoten der Grad der Bildung abgeleitet werden. Beide Verfahren sind kritisch zu betrachten.

Peter R. Hofstätter schreibt in seinem 1957 erschienenen Fischer Lexikon Psychologie: *»Unter den Begabungen und Eigenschaften des Menschen nimmt die Intelligenz in zweifacher Hinsicht eine ausgezeichnete Stellung ein: Sie gilt der psychologischen Spekulation schon von alters her als eine besonders hoch zu bewertende, in einem eigentlicheren Sinne als andere ›göttliche‹ Fähigkeit, und sie ist zugleich die Eigenschaft, um deren Messung sich die empirische Psychologie der letzten 75 Jahre am nachdrücklichsten bemühte.«*

Mit dem ›göttlichen Funken‹ beginnt auch die Ausführung zum Stichwort Intelligenz im Philosophischen Wörterbuch des Körner Verlages, Ausgabe 1969: *»... die dem Menschen eigentümliche geistig-verstandesmäßige Begabung«*. So eigentümlich scheint die intelligente Begabung des Menschen jedoch gar nicht zu sein. Verhaltensforscher sind jedenfalls durchaus geneigt, auch Tieren intelligente Leistungen zuzugestehen.

Setzt man ein Huhn auf einen rechteckigen Platz, der an drei Seiten von einem Zaun begrenzt ist, und stellt hinter dem Zaun einen Freßnapf auf, dann flattert das dumme

Huhn lediglich aufgeregt gegen den Zaun und läßt dabei reichlich Federn. Schimpansen hingegen sind nicht nur in der Lage, eine außer Reichweite vor dem Käfig liegende Banane mit einem Stock zu angeln, sie sind auch clever genug, zwei Teile einer Stange zusammenzustecken, wenn jedes einzelne Teil zu kurz ist, um den Leckerbissen der Verdauung zuzuführen.

Sind Schimpansen und Ratten intelligent?

Besonders brave und für Laborversuche beliebte Schüler sind Ratten. Wenn es darum geht, den Bauch mit Leckereien zu füllen, finden sie nicht nur den Weg durch ein Labyrinth zu einem Stück Käse, sie prägen sich diesen Weg sogar ein, lernen also. Verwirrend wird das intelligente Spiel mit den Versuchstieren, wenn uns berichtet wird, man könne einer Ratte bestimmte Reaktionen antrainieren, ihr Gehirn sodann an andere Ratten verfüttern, die daraufhin die gleichen Fähigkeiten zeigen, ohne sie jemals gelernt zu haben. Haben die Kannibalen doch recht?

Möglicherweise wurde die Erforschung der Intelligenz dadurch erschwert, daß sie für die menschliche Kultur im allgemeinen und jedes einzelne Individuum im besonderen eine so große Bedeutung hat. Dieser Umstand mag uns für eine objektive Betrachtung blind gemacht haben. Erst jetzt, durch die Projektion in den Computer, kann man sich der theoretischen Durchdringung des Phänomens Intelligenz nicht mehr entziehen.

Noch einmal Hofstätter: »*... unter einigermaßen normalen Verhältnissen der Lebenserfolg des Individuums in seiner Kultur und dessen Beitrag zu deren Wahrung und Mehrung von gedanklichen Leistungen stärker abhängt als etwa von der Entfaltung physischer Kräfte und spezieller Fähigkeiten... Vielfach wird auch die ethische Höhe einer Persönlichkeit der Vollkommenheit ihres Wissens um Gut und Böse gleichgesetzt. Aus diesen zum Teil schon dem vorwissenschaftlichen Denken entstammenden Annahmen erklärt sich die Tendenz, jede Form der sozialen Bevorzugung mit der Zuschreibung höherer Intelligenz zu koppeln; geringer geachtete Stände und Minoritäten, mißliebige Personen, Angehörige als fremd empfundener Rassen, Kriminelle, sogar Frauen im Gegensatz zu Männern und nicht zuletzt der Teufel des Volksglaubens gelten daher als weniger intelligent, die Hochgeachteten und Anerkannten hingegen als ›die Intelligenz‹ schlechthin.*«

Intelligenz = soziale Stellung?

Der menschliche Geist scheint ein natürliches Bedürfnis zu haben, sich ein Weltbild zu schaffen und mit einer Mitte auszustatten. Wer nur ein bißchen Lebenserfahrung hat, den wundert es nicht, daß der Mensch selbst und alles, was sein ist, diese Mitte besetzt.

Die Entwicklung der Wissenschaft verlief auf einem dornenreichen Weg, auf dem der Mensch immer mehr von seiner selbstherrlichen Mitte aufgeben mußte. Für die Verbreitung der Erkenntnis, daß nicht die Erde, also die Heimat des Menschen, sondern die Sonne der Mittelpunkt »der Welt« sei, mußten kühne Geister ihr Leben lassen.
Vielleicht setzt die KI-Forschung den Anfang einer kopernikanischen Wende für das Intelligenzbild. In der Vergangenheit wurde Intelligenz allgemein als eine menschliche Fähigkeit und mehr oder weniger unteilbare Größe betrachtet. Der Intellekt ist der eigentliche Mittelpunkt des heutigen Weltbildes. Als Agens in Wissenschaft und Technik bestimmt und verändert er die Welt. Damit ging eine Wandlung auch in der ethisch-moralischen Weltanschauung einher, die sich, vereinfacht gesagt, von der christlich-religiösen gelöst und stetig entfernt hat. Wenn es aber richtig ist, daß Intelligenz der Mittelpunkt unseres Weltbildes ist, dann wundern die heftigen Auseinandersetzungen nicht, die entstehen, weil dieser Mittelpunkt nun von seinem Thron gerissen werden soll, indem er in künstlicher Form beliebig reproduzierbar werden soll.

Die Suche nach dem Intelligenzatom

Um Intelligenz wissenschaftlichem Verstehen aufzuschließen, muß sie vor allem (objektiv) meßbar gemacht werden. Das heißt, sie auf ihre kleinste Einheit zurückzuführen. Intelligenz offenbart sich im Erzeugen von Ordnung, in Einteilung nach richtig und falsch, im Aufbau von Strukturen. Dabei ist das ordnende Element die Richtung, so daß Richtung als kleinstes, unteilbares Maß für Intelligenz betrachtet werden kann. Durch die fortschreitende Überlagerung von Richtungen entstehen Strukturen. Diese Betrachtungsweise scheint jedoch zumindest im gegenwärtigen Stadium der Entwicklung zu elementar zu sein, um das Verständnis der hochkomplexen Vorgänge, mit denen sich die KI beschäftigt, zu erleichtern.
Der Mensch bewegt sich in einem Lebensraum, der mit dem Begriff Natur gefaßt wird. Die Vorgänge in der Natur, von der subatomaren bis zur astronomischen Ebene, unterliegen Gesetzen, die, ohne unbedingt einen religiösen Anspruch damit zu verbinden, als intelligent bezeichnet werden müssen. Der forschende Geist macht sich ein Bild von der Natur, indem er ihre Gesetze in seine eigene Sprache zu übersetzen versucht und das so gewonnene Modell an der Wirklichkeit überprüft. Forschungsergebnisse sind in dem Maße richtig und bedeutend, wie sie Natur abbilden und damit gezielte Eingriffe ermöglichen. Das Besondere der KI-Forschung

liegt darin, daß nicht ein Modell bestimmter Aspekte der Natur, also ein physikalisches, chemisches oder ein ethisches, gefunden werden soll, sondern ein Modell der Fähigkeit, Modelle zu finden.
Die wesentlichen Arbeitsschritte in diesem Prozeß sind in dem Wort Intelligenz bereits gefaßt, das vom lateinischen »intellegere« abgeleitet ist und mit »erkennen«, »unterscheiden«, »einsehen« übersetzt werden kann. Zuerst muß der Gegenstand der Untersuchung in seinen Eigenschaften erkannt, also wahrgenommen werden. Soweit die bloße Anschauung nicht ausreicht, wird er nach dem Prinzip von Versuch und Irrtum verschiedenen Einflüssen ausgesetzt. Die so gewonnenen Erfahrungen werden geordnet, also voneinander unterschieden. Sie ergeben ein erstes Modell des Gegenstandes. Aus diesem Modell, das den Gegenstand auf die für den Erkenntniszweck wichtigen Elemente reduziert, werden Einsichten gewonnen, die danach als Grundlage für weitere Untersuchungen dienen, also die Wahrnehmung auf bestimmte Aspekte konzentrieren und den Unterscheidungsprozeß strukturieren, wobei die gewonnenen Erkenntnisse immer wieder überprüft werden. Dabei wird ein fortschreitend zutreffenderes Modell des Erkenntnisgegenstandes gedacht. Die Erkenntnisschritte werden gelernt und gegebenenfalls bei künftigen Erkenntnisversuchen herangezogen.

Der Prozeß des Gewinnens von Erkenntnissen

Die KI-Forschung ist heute noch weit davon entfernt, diesen Erkenntnisprozeß maschinell zu reproduzieren und einen Rechner zu befähigen, selbständig Einsichten zu gewinnen. Tatsächlich scheinen die Probleme schon im ersten Stadium der Erkenntnis, der Wahrnehmung, fast unüberwindlich. Das Kapitel über Mustererkennung beschäftigt sich ausführlicher mit dieser Frage.
Die intelligente Fähigkeit, neue Erkenntnisse zu gewinnen, liegt heute noch weit hinter dem Horizont der KI-Forschung. Das Programm MACSYMA zum Beispiel ist zwar in der Lage, durch Anwendung mathematischer Axiome Lehrsätze der Gruppentheorie, der Topologie, der Geometrie, der Mechanik etc. zu beweisen. Es rekonstruiert jedoch lediglich bekannte, längst bewiesene Sätze auf dem Niveau von Vordiplomaufgaben.
Jahrzehntelang wurden die Bemühungen um künstlich geschaffene Intelligenz in der Öffentlichkeit wenig beachtet. Zum journalistischen Thema wurden sie, als ein bescheidener Teilerfolg erste marktfähige Produkte hervorzubringen begann: Expertensysteme.

Systeme für Experten

Es gibt viele überzeugende Gründe zu bezweifeln, ob Maschinen jemals in menschlichem Sinne intelligent sein werden, sich also in beliebigen Situationen zurechtfinden können. Der Hausroboter, der uns nicht nur morgens weckt und bereits den Frühstückstisch gedeckt hat, sondern der nebenbei auch noch ein unterhaltsames Gespräch über »Gott und die Welt« mit uns führt, wird noch lange auf sich warten lassen.
Wird der potentielle Einsatzbereich auf ein klar umrissenes Gebiet begrenzt, steigt die Wahrscheinlichkeit, daß Maschinen selbständig qualifizierte Leistungen erbringen können. Doch obwohl bei der Wahrnehmung, der Mustererkennung, schon interessante Fortschritte gemacht wurden, ist die Brücke vom bloßen Erkennen zur Erkenntnis noch nicht geschlagen.
Eine Forschergruppe der Universität Hamburg ist mit der Entwicklung eines Programms beschäftigt, das die Form bewegter Objekte erkennt. Computer arbeiten schon lange erfolgreich mit Videokameras zusammen, die das aufgenommene Bild bekanntlich in einzelne Punkte bestimmter Farbwerte auflösen, es digitalisieren. Diese Bilddaten lassen sich von einem Rechner leicht verarbeiten. Aus einer Folge von Einzelbildern werden durch Vergleich alle Teile ermittelt, die sich verändert haben, und aus diesen Anhaltspunkten wird die äußere Form des sich bewegenden Objekts abgeleitet. Da Bilder einen extrem hohen Informationsgehalt haben und hier obendrein eine Folge von Bildern zu verarbeiten ist, braucht das Programm schon sehr viel Zeit, wenn es nur die großen Umrisse errechnen soll. Immerhin hat man einen Ansatz gefunden, aus bewegten, zweidimensionalen Bildern das Modell eines Gegenstandes herauszufiltern und in Werte umzusetzen, die ihn in seiner Dreidimensionalität beschreiben.

Mustererkennung: Trotz vieler Fortschritte noch immer ein Problem

Praktikablere Erfolge erzielen die KI-Forscher jedoch dort, wo sie sich bemühen, die einfache Datenverarbeitung intelligenter zu machen. Ein Beispiel soll zeigen, was damit gemeint ist.
In der traditionellen Datenverarbeitung werden Einzeldaten miteinander verknüpft. Um zum Beispiel das Alter einer Person zu erfahren, muß ich ihren Namen, ihre Kundennummer oder irgendeinen anderen Schlüssel kennen. Im Prinzip funktioniert das nicht anders als bei

einem Telefonbuch. Suche ich eine bestimmte Rufnummer, muß ich wissen, unter welchem Stichwort der Teilnehmer eingetragen ist. Selbst wenn ich den Namen des Teilnehmers kenne, finde ich seine Telefonnummer nicht unbedingt, weil ich nicht weiß, in welcher Form er alphabetisch eingeordnet wurde. Bei Firmen oder Behörden erlebt man das recht oft. Suche ich die XY-Schule, muß ich, je nach Telefonbuch, unter Behörden, Stadtverwaltung oder Schulen nachschauen, nicht unter XY.

Suchaufgaben:
Eine typische KI-Anwendung

Eine intelligente Datei kann durch einkreisende Fragen bei der Suche behilflich sein. Sie könnte Geschlecht, Beruf, Alter, Wohnort und andere Kriterien abfragen, um schließlich eine Antwort zu geben, die, je nach der Qualität und Vollständigkeit der erfolgten Eingaben, eine einzige in Frage kommende Rufnummer nennt und dazu angibt, mit welcher Wahrscheinlichkeit es die gesuchte ist — oder eine Liste aller durch den Dialog eingekreisten Nummern ausgibt.

Um diese Leistung erbringen zu können, muß das System ein bestimmtes Wissen haben, das es ihm ermöglicht, die einkreisenden Fragen zu stellen, die eingegebenen Antworten zu bewerten, weiterführende Fragen abzuleiten und schließlich ein Ergebnis oder eine Prognose auszugeben. Es geht also um eine Entwicklung von der reinen Datenverarbeitung zur Wissensverarbeitung.

Ist das zu bearbeitende Feld begrenzt, kann das notwendige Wissen durch Befragung von Experten ermittelt und auf beschreibbare Regeln zurückgeführt werden. Mit Hilfe dieser Wissenselemente kreist das Programm die Frage des Benutzers ein. Wenn die gesuchte Person männlich ist, fallen alle Telefonteilnehmer mit weiblichen Vornamen fort. Hat die gesuchte Person einen Doktortitel, handelt es sich aufgrund der und der Tatsache mit einer Wahrscheinlichkeit von x% um einen Arzt usf.

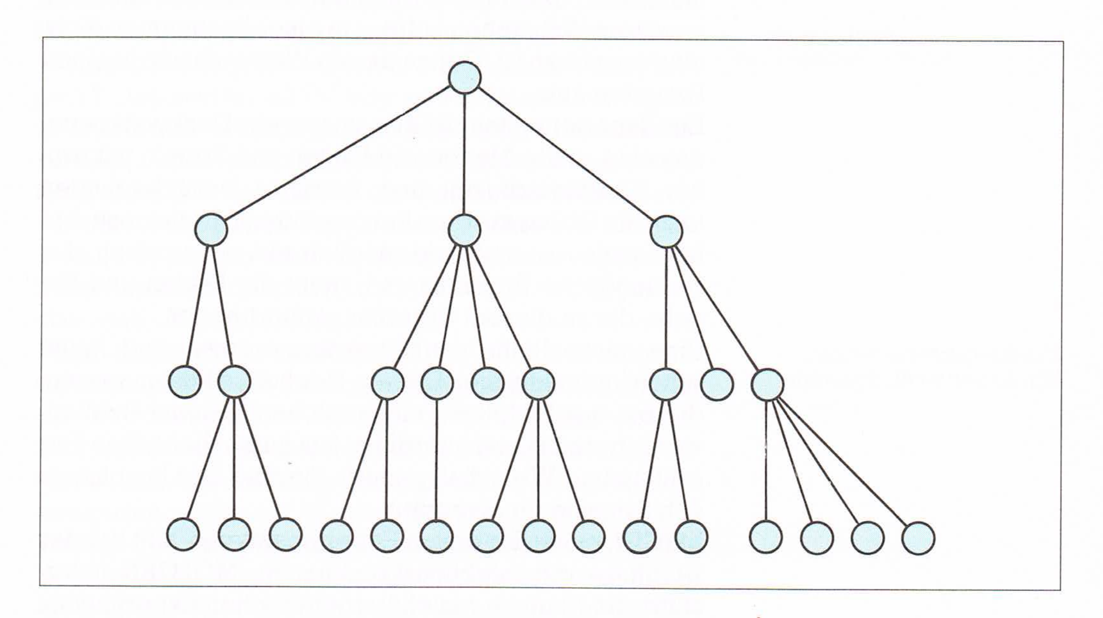
Grundkomponenten eines
Expertensystems

Ein Expertensystem besteht im allgemeinen aus fünf Komponenten. Es verfügt über eine Wissensbasis (**knowledge base**), in der Fakten und Regeln erfaßt sind. Die Dialogverwaltung (**dialog management**) unterhält die Kommunikation mit dem Benutzer. Der Schlußfolgerungsmechanismus (**inference machine**) sucht nach Fakten und Regeln und verknüpft sie nach vorgegebenen Strategien. Die Erklärungskomponente (**explanation component**) dokumentiert dem Benutzer, unter Einfluß welcher Fakten und Regeln ein Ergebnis zustande gekommen ist. Und die Wissenserwerbung (**knowledge acquisition**) ermöglicht es dem System,

kontinuierlich zu lernen, also neue Fakten und Regeln aufzunehmen und in die übrigen vier Komponenten einfließen zu lassen.

Ein wesentliches Merkmal von Expertensystemen besteht darin, daß sie nicht nach strengen Algorithmen eine Lösung errechnen. Hinter dem Begriff **Algorithmus** verbirgt sich eine Prozedur, die für alle im Rahmen einer Aufgabe möglichen Situationen eine definitive Handlungsanweisung bereithält. Bei der Bearbeitung eines vielschichtigen Problems sind jedoch so viele Entscheidungen zu treffen, daß sich die Menge der zu untersuchenden Möglichkeiten explosionsartig vergrößert.

Ausgehend von der Problemstellung gabelt sich der Lösungsweg bei jeder zu treffenden Entscheidung. Nach wenigen Schritten kann der Entscheidungsbaum schon hundertfach verzweigt sein. Bei Anwendung eines Algorithmus könnte das Programm das Problem nur stur Pfad um Pfad untersuchen, bis es früher oder später eine Lösung findet, und es müßte alle Pfade durchforsten, um mit Gewißheit die beste Lösung herauszurechnen.



Jeder Handlungsablauf setzt sich aus einer Vielzahl von Entscheidungen zusammen, die in einer baumartigen Struktur übersichtlich geordnet werden können.

Das Aufregende an der KI-Forschung ist, daß sie uns zwingt, über unser eigenes Denken nachzudenken. Offensichtlich berücksichtigt unser Gehirn nicht alle möglichen Lösungswege, wenn es ein Problem bearbeitet.

Menschliche Intelligenz findet Wegweiser im Gedankenwirrwarr

Dann würden wir nämlich morgens gar nicht aus dem Bett finden, weil wir bis zum Abend damit beschäftigt wären, nach der optimalen Lösung für dieses Problem zu suchen. Die menschliche Intelligenz verfügt über Methoden, in das unüberschaubare Gewirr eines Entscheidungsbaumes Wegweiser zu setzen und damit den Suchvorgang entscheidend zu begrenzen.
Heuristische Methoden (**Heuristik**) führen nicht immer wie ein Algorithmus mit logischer Konsequenz zu dem einzig möglichen, richtigen Ergebnis. Für deren Richtigkeit besteht manchmal nur eine gewisse Wahrscheinlichkeit. Andere weisen in eine Lösungsrichtung, die sich am Ende durchaus als Sackgasse entpuppen kann. Bei Anwendung der heuristischen Methode werden die einzelnen Etappen auf dem Weg zur Lösung fortlaufend bewertet, und nur der Pfad mit dem jeweils besten Zwischenergebnis wird weiter verfolgt. Erst wenn sich diese Richtung als Holzweg zeigt, werden andere Pfade untersucht. In den Bewertungsprozeß können Informationen aus der Wissensbasis einfließen. Aus ihnen kann geschlossen werden, daß bei einer bestimmten Konstellation von Fakten die Lösung eines Problems mit einer gewissen Wahrscheinlichkeit in einer bestimmten Richtung zu suchen ist. Genau dieses Wissen macht ja einen Experten aus.
Ein Expertensystem ist also zuerst ein Denkwerkzeug, das eine große Menge von Fakten und Regeln mit großer Geschwindigkeit und Präzision berücksichtigen kann. Im Dialog mit dem Benutzer kreist es die gesuchte Information so weit wie möglich ein, es ermittelt das bestmögliche Ergebnis und nennt die Fakten und Regeln, die zu diesem Ergebnis geführt haben.

Expertensysteme heute: Von Experten für Experten

Eine ganze Reihe von Expertensystemen sind heute schon im praktischen Einsatz. Der hohen Kosten wegen, die bei der Erstellung eines solchen Systems anfallen, werden sie bislang allerdings fast ausschließlich in Forschung und Wirtschaft genutzt. Sie sind also hauptsächlich Systeme für Experten.
MYCIN, eines der ersten Expertensysteme, hilft bei der Diagnose von Infektionskrankheiten. **MOLGEN** unterstützt die Planung molekulargenetischer Experimente. **R1** und **Conad** werden bei der Entwicklung von Computersystemen eingesetzt.
Die Tendenz geht heute dahin, Expertensysteme für alltäglichere Einsatzgebiete zu entwickeln, die Laien Expertenwissen zur Verfügung stellen. Eines der großen Hindernisse ist dabei noch die Bedienung der Anlagen. Deshalb bemüht man sich besonders um die Bedienerfreundlichkeit.

Kritik an Expertensystemen ...

Während die Industrie behauptet, die künstliche Intelligenz sei da, weil es Expertensysteme gibt, sagen Kritiker wie der Psychologe und KI-Experte Roger Schenk von der Yale-University, die Expertensysteme seien schlechter, als von den Anbietern behauptet wird. Jeder würde sich lieber von einem Arzt mit langjähriger Berufserfahrung behandeln lassen als von einem Medizinstudenten, der alle Regeln und Fakten kennt. Ein Programm, das nur aus Regeln bestehe, könne sich nicht weiterentwickeln und wirklich dazulernen.
Hubert Dreyfus hält das Grundprinzip, nach dem Expertensysteme arbeiten, für verfehlt. Menschen würden nicht nach strengen Regeln handeln. Sie erkennen Situationen als Ganzes und vergleichen sie mit früheren Situationen, in denen sie erfolgreich gehandelt haben. Wenn man also einen Experten nach seinen Regeln fragen würde, hätte dies lediglich zur Folge, daß man ihn zwingen würde, sich welche auszudenken oder sich an Anfängerregeln zu erinnern. Anfänger würden in der Tat Regeln benutzen.

... und ein uralter philosophischer Streit

Hinter dieser Argumentation verbirgt sich ein philosophischer Theoriestreit. Auf der einen Seite stehen mit Sokrates, Descartes, Kant und Hussel bis hin zum frühen Wittgenstein die Verfechter der Annahme, alles ließe sich formalisieren, alles ließe sich auf objektive Eigenschaften und formallogische Beziehungen zurückführen. Auf der anderen Seite stehen Denker wie Aristoteles, Pascal und Heidegger, die Gegenstände und Sachverhalte als Ganzheit beschreibbar verstehen. Dreyfus meint, der seit nunmehr zweitausend Jahren währende Streit der Philosophen würde im Konzept der KI seinen objektiven Testfall finden.
Mehr praktisch orientierte Kritiker sind der Ansicht, daß unsere Zeit von berechnenden Gedanken geprägt sei. Expertensysteme würden diese Haltung verstärken, wenn man glaube, daß sie funktionieren. Abgesehen davon, daß es ein Erfahrungswissen gäbe, ein Fingerspitzengefühl, ein Wissen, das sich nicht einfach in Wörter und Regeln fassen ließe, sogenanntes graues Wissen, würde der Kommunikationsprozeß unter den Menschen weiter reduziert und noch mehr auf die Maschine konzentriert. Gerade im gemeinsamen Gespräch mit anderen Personen würden aber auch neue Ideen geboren. Ein Expertensystem könne nur immer wieder die alten Antworten geben.
Mancher Sympathisant der KI-Bewegung macht sich weniger Gedanken. Die Frage: »Können Computer denken?« wird dann gern mit der Gegenfrage beantwortet: »Warum nicht?«

Kreative Intelligenz

Auf der Suche nach der künstlichen Intelligenz stellt sich die Frage, ob es nicht auch eine künstliche Kreativität geben kann. Beide Fähigkeiten des Menschen ergänzen sich; die eine scheint ohne die andere weniger Wert zu haben. Und dennoch liegen sie in fortwährendem Wettstreit miteinander.

Die Methode der Intelligenz, vertreten durch die Zunft der Wissenschaftler, ist die Analyse, die zerteilt und gliedert, um aufzudecken, was die Welt im Innersten zusammenhält. Vornehmlich zuständig für Kreativität ist die Gilde der Künstler, die durch Synthese bemüht sind, sich ein ganzheitliches Bild von der Welt zu schaffen. Vereinfacht gesagt entdeckt der Wissenschaftler, indem er das Ganze in Teile zerlegt; der Künstler schöpft, indem er Teile zu einem Ganzen vereint.

Es gibt nichts Neues unter der Sonne

So wenig der Intellekt Teile erforschen kann, die das Ganze nicht enthält, kann die Kreativität aus Teilen kein Ganzes schöpfen, das es nicht gibt. Beide finden nur, jedes auf seine Art, was vorher versteckt, aber eben doch vorhanden war.

Wenn der Intellekt Wörter untersucht, so stellt er fest, daß sich alle aus einer begrenzten Menge von Lauten zusammensetzen, die der Einfachheit halber einmal mit Buchstaben gleichgesetzt werden sollen. Bei näherer Betrachtung der Materie stellt sich heraus, daß Buchstaben nicht beliebig zusammengefügt werden können, wenn ein aussprechbares Wort entstehen soll. In der deutschen Sprache verlangt ein »Q« immer die Gefolgschaft eines »U«. Andere Buchstaben dulden nur manche Kollegen hinter sich. So verweigert das »G« eine Verbindung mit »B«, »C«, »D«, »F« usf. Manche Buchstaben schließen sich zu Cliquen zusammen: »CH«, »CK«, »SCH«, um nur einige zu nennen. Und nie stehen mehr als vier Konsonanten in einer Reihe: »SCHL«.

So ließe sich eine Verknüpfungstafel erstellen, die alle möglichen Kombinationen von Konsonanten und eine Liste der verbindenden Vokale enthält, zu denen auch die umgelauteten (»Ä«, »Ö«, »Ü«), die zusammengesetzten (»AU«, »ÄU«, »EI« etc.) und gedehnten (»AA«, »AH« etc.) genommen werden müßten.

Nach dieser sorgfältigen Analyse können durch wechselndes Zusammenfügen von Konsonantenkombination und Vokalverbindung beliebig Kunstwörter produziert

werden. Allerdings wurden nicht wirklich alle Regeln erschöpfend entdeckt und berücksichtigt, und so kann es schon mal zu einem Kunstfehler kommen. Nach dem skizzierten Rezept wäre zum Beispiel die Buchstabenfolge PAHSTEI ein mögliches Ergebnis. Sie widersetzt sich der deutschen Zunge zwar nicht, doch gibt die Muttersprache unserer Väter die Folge HST nur her, wenn zwei Wörter zusammengesetzt werden: NAHSTRECKE.

Kreativität: Schöpfen aus der Ursuppe

Die Kreativität würde ganz anders vorgehen. Sie schöpft einen gehörigen Löffel aus der Buchstabensuppe und schiebt die Lettern hin und her, bis ein brauchbares Wort entstanden ist. Das eigentlich Kreative besteht ja darin, neue Verbindungen von Teilen zu finden, also ein Wort, eine Folge von Buchstaben, die bislang noch nicht bekannt war, vielleicht, um sie als Namen für ein neues Produkt zu verwenden. Auch die Kreativität muß sich an die vorgefundenen Regeln halten, wenn das Wort aussprechbar sein soll, aber sie muß die Regeln nicht der Analyse unterziehen. Sie fügt mit schöpferischer Urkraft zusammen, ohne sich viel Kopfzerbrechen über das Wie und Warum zu machen. Deswegen erfindet sie vielleicht auch ein Wort wie KHART, das mit der Buchstabenfolge KH den Regeln eigentlich widerspricht, dennoch aussprechbar ist und möglicherweise dem Zweck besonders gerecht wird, für den es gesucht wird.

Um Neues zu schaffen, sind beide notwendig: Kreativität und Intelligenz. Die Kreativität kann als Energie betrachtet werden, die eine beliebige und chaotische Vielfalt von Möglichkeiten hervorsprudelt wie eine Fontäne. Die Intelligenz (Richtung) lenkt diesen Strom in geordnete Bahnen, und zwar auf der Grundlage der Gesetze, die sie erkannt hat und die sie deshalb berücksichtigen kann.

Über chaotische Energie verfügt BASIC durch die Zufallszahlen, welche die Funktion RND erzeugt. Die ordnende Intelligenz fügt der Programmierer in Gestalt von BASIC-Zeilen, welche die Regeln enthalten, hinzu. Ein auf diese Weise strukturiertes Programm, das Kunstwörter erzeugen soll, kann zum Beispiel so aussehen:

```
1 REM KUNSTWRT.DEM
10 DIM k$(14),v$(11):RANDOMIZE TIME
20 FOR j=0 TO 14:READ k$(j):NEXT j
30 FOR j=0 TO 11:READ v$(j):NEXT j
100 w$=""
110 l=INT(RND(1)*3)*3
120 FOR j=0 TO l
```

```
130 xk=INT(RND(1)*15)
140 xv=INT(RND(1)*12)
150 x$=k$(xk)
160 IF (j-1)/3=INT((j-1)/3) THEN x$=v$(x
v)
180 w$=w$+x$
190 NEXT j
210 xv=INT(RND(1)*12)
220 x$=v$(xv):w$=w$+x$
230 xk=INT(RND(1)*15)
240 x$=k$(kv):w$=w$+x$
300 PRINT w$
310 IF INKEY$="" GOTO 310
320 GOTO 100
330 DATA B,D,F,G,K,L,M,N,P,R,S,T,V,W,Z
340 DATA A,AU,AA,E,EI,EU,EE,I,IE,O,OO,U
```

KUNSTWRT.DEM arbeitet nach einer vereinfachten Regel, um lesbare Phantasiewörter aus bestimmten Konsonanten und Vokalen zu produzieren. So werden zum Beispiel nur bestimmte Konsonanten und Vokale verwendet.

Beschreibung des Programmes KUNSTWRT.DEM

In **Zeile 10** werden die Feldvariablen k$ und v$ dimensioniert und aus dem Timer ein Anfangspunkt für die Pseudozufallszahlenreihe gesetzt. **Zeile 20** liest die Konsonanten in die Variable k$(j), **Zeile 30** die Vokale in v$(j) ein, die in den **Data-Zeilen 330 und 340** abgelegt sind. Die Stringvariable w$ soll das Kunstwort aufnehmen. In **Zeile 100** wird sie geleert.

Durch die zweite Vereinfachung der angewendeten Wortschöpfregel wird bestimmt, daß jedes Wort aus der immer gleichen Folge von Konsonanten (K) und Vokalen (V) zusammengesetzt wird, und zwar in dieser Form: K-V-K-K-V-K-K-...V-K. Dabei kann unberücksichtigt bleiben, ob die entstehenden K-K-Verbindungen aussprechbar sind, weil jede K-V-K-Kombination aus den verwendeten Buchstaben als Silbe lesbar ist. Also muß auch ein daraus zusammengesetztes Wort als Folge von K-V-K-Silben lesbar sein.

Die **Zeile 110** ordnet der Variablen 1 den Zufallswert 0, 3 oder 6 zu und bestimmt damit eine vorläufige Wortlänge von 1, 4 oder 7 Buchstaben (oder Doppelvokalen: »AU«, »AA«, »EI«, »EE«, »IE«, »OO«), wenn 1 in **Zeile 120** den Zielwert für den Schleifenzähler j bestimmt, da j mit dem Wert 0 zu zählen beginnt.

Nachdem in **130 und 140** entsprechende Zufallswerte für xk und xv ermittelt sind, bestimmt xk in **Zeile 150** einen Konsonanten, der vorübergehend x$ zugeordnet wird.

Die Bedingung in **Zeile 160** sorgt dafür, daß beim zweiten, fünften und siebten Durchlauf der Variablen x$ ein

Vokal zugeordnet und der vorher gefundene Konsonant durch ihn überschrieben wird.
In **Zeile 180** wird der zufällig gefundene Buchstabe an den String w$ angehängt.
In den **Zeilen 210 bis 240** wird w$ abschließend noch um einen Vokal und um einen Konsonanten verlängert, bevor **Zeile 300** das Kunstwort, dessen Elemente in w$ zusammengestellt wurden, auf den Bildschirm schreibt.
Die **Zeile 310** fängt das Programm in einer Schleife, bis eine beliebige Taste gedrückt wird. Danach beginnt mit Rücksprung nach **100** ein neuer Durchlauf.
Zugegeben, Wortschöpfungen wie ZEILZIEB, RUMLOOB oder RAGLAB sind nicht besonders überwältigend. Auch ein gelegentliches ZAUN oder WEIBER macht dieses kleine Demonstrationsprogramm nicht besonders beeindruckend. Doch kann auf diese Weise mühelos eine beliebig lange Liste von Wörtern erzeugt werden. Auf ähnliche Weise ist der Firmenname EXXON erfunden worden. Aus einer langen Liste von Zufallswörtern wurden einige wenige gewählt. Sie wurden in zahlreichen psychologischen Tests und Umfragen erprobt, bevor die endgültige Entscheidung fiel.

Kann denn Liebe künstlich sein?

Uns're Umarmung der Blume

Der zärtliche Wind stillt die Blüte uns'rer Jahre
Rund atmet eine Süße
Ein unendliches Meer belebt das Glänzen Deiner Haare
Befreite, befreite Blüte

Uns're freie Erfüllung stillt die Tiefe Deiner Augen
Erhaben höht euer Gebinde
Der befreite Hauch träumt den Saft von schwarzen Trauben
Unendliche, lichte Linde

Eure erhabene Lust glüht die Weichheit Deiner Lippen
Frisch fliegt uns're Lenze
Jede beschwingte Stille lebt die Süße dran zu nippen
Duftende, zärtliche Gänze

Manch' beschwingte Lust stillt was früher ich nie fand
Unendlich strömt manch' Küssen
Jede schöne Erfüllung lebt mein Herz in Deiner Hand
Heiliges, rundes Wissen

Seit den Experimenten der Surrealisten ist die Bereitschaft eines größeren Leserkreises gestiegen, die Verse dieses Gedichts auf sich wirken zu lassen und sie als Ausdrucksform ernst zu nehmen. An dieser Stelle in einem Buch über künstliche Intelligenz eingerückt, schwant dem Leser allerdings wohl schon, was da auf ihn losgelassen wird.
Die surrealistischen Schriftsteller, allen voran André Breton, haben das automatische Schreiben propagiert. Dabei soll, frei von individueller und kollektiver Kontrolle und unter Ausschaltung des ordnenden Intellekts, dem psychischen Mechanismus die Möglichkeit des ungebundenen Ausdrucks gegeben werden, um die hinter der scheinbaren Realität stehenden seelischen Bezüge aufzudecken.
Sieht man einmal davon ab, daß die Surrealisten, beeinflußt von Sigmund Freud, eine allgemein verbindliche tiefenpsychologische Realität voraussetzen, so arbeitet das Lyrikprogramm LOVEPOEM.ART durchaus nach dem Prinzip des automatischen Schreibens, denn jede inhaltliche Bewertung der entstehenden Satzgebilde unterbleibt. Der Aspekt ordnender Intelligenz beschränkt sich auf die Einhaltung grammatikalischer Strukturen der deutschen Sprache und auf ein vorgegebenes Versschema.

»Automatisches« Schreiben: Wo bleibt die (ordnende) Intelligenz?

Es ist nicht zu leugnen, daß jeder Mensch eine psychische Realität in sich birgt und diese mit geeigneten Mitteln wie dem Traum, der Trance oder eben dem automatischen Schreiben und Malen zum Ausdruck bringen kann. Ob diese seelische Realität aber allgemein verbindlich, also auf beliebige Individuen übertragbar ist, bleibt zumindest für Form und Inhalt surrealistischer Ästhetik fraglich. Diese Übertragbarkeit würde nämlich verbindliche Regeln voraussetzen. Die einzige Regel für die surrealistische Tätigkeit und womöglich ebenso für den Traum und andere psychische Vorgänge besteht aber darin, daß es keine formalen Regeln gibt. Alle Inhalte können beliebig assoziiert werden. Die Surrealisten haben das selbst in der Definition treffend ausgedrückt. Surrealismus sei das zufällige Zusammentreffen eines Regenschirms mit einer Nähmaschine auf einem Seziertisch.
Jede ungewöhnliche Kombination von (ästhetischen) Elementen vermittelt den Eindruck einer tieferen Bedeutung. Da die Elemente und ihre Kombination beliebig sind, ist aber auch die als tieferliegend vermutete Bedeutung beliebig. Die Surrealisten haben möglicher Kritik vorgebeugt, indem sie reklamierten, ihre Werke hätten keine Bedeutung. Das ändert aber nichts an der

Tatsache, daß surrealistische Werke dem Betrachter bedeutsam erscheinen. Der Begriff Surrealismus (französisch: Überwirklichkeit) selbst bezieht sich auf dieses Phänomen.

Surrealistische Gedichte per Programm

LOVEPOEM.ART berücksichtigt die Regeln surrealistischer Literaturproduktion. Der Unterschied besteht lediglich im Fehlen einer individuellen Psyche, die aber nur eine subjektive Auswahl aller möglichen zufälligen Verbindungen von Inhalten hervorbringen würde. Je größer das interne Wörter- und Phrasenbuch des Programmes angelegt wird und je mehr syntaktische Strukturen vorgesehen werden, desto umfassender ist das Ausdrucksvermögen eines solchen »Sur«-Programms.

Das Programm besteht aus vier Komponenten. Die Basis bildet das Lexikon, das Wörter, Phrasen, Reimpaare und Endungen enthält. Dieses Lexikon kann beliebig erweitert oder geändert werden. Die vorliegende Zusammenstellung des Wörtermaterials erfolgte unter dem Aspekt Liebe. Die individuelle Psyche des Programmierers filtert also Elemente aus dem zur Verfügung stehenden Wortschatz heraus und geht somit in das lyrische Endprodukt ein.

Die zweite Komponente ersetzt die subjektive Intuition des Dichters durch zufällige Auswahl der Begriffe per BASIC-Befehl RND aus dem Lexikon.

Die dritte Komponente strukturiert die vom Lexikon vorgegebenen und durch die Random-Intuition ausgewählten Wörter zu einem Text, der den grammatikalischen Gesetzen der deutschen Sprache angepaßt ist. Das Endprodukt ist ein Gedicht mit einer Überschrift, das aus Vierzeilern besteht, die alle dem Reimschema A-B-A-B folgen. Die vorgegebene Struktur ist verbindlich im Programm festgeschrieben. Es ist mit überschaubarem Aufwand auch möglich, verschiedene Reimschemata, Versstrukturen und Strophenzahlen bereitzuhalten und vom Programm jeweils neu per Zufall auswählen zu lassen. Hierauf wurde deshalb verzichtet, weil das Programm dann noch umfangreicher geworden wäre, ohne daß dadurch eine entsprechend bessere Einsicht in das Arbeitsprinzip gewonnen würde.

Bauelemente künstlicher Lyrik

Als Bauelemente für die künstliche Lyrik wurden verwendet: Attribute (Pronomen, bestimmte und unbestimmte Artikel wie mein, der, ein) und Endungen (-e, -er, -es). Beide sind vom Genus (Geschlecht) des jeweiligen Substantivs (Seele, Herz, Freude) abhängig. Die Substantive sind deshalb im Lexikon durch ein vorangestelltes »m«, »f« oder »n« klassifiziert. Die Deklination der Adjektive (schön, hell, licht) erfolgt durch Anhängen der Endung, die durch das Genus des Substantivs be-

stimmt ist: schön-e Seele. Als Prädikat werden einfache Verben verwendet (streichelt, küßt, haucht). Objekte sind Verbindungen aus Akkusativ- und Genitivobjekt, teilweise in erweiterten Formen (den Horizont der Stille, das warme Licht von Kerzen, die Blüte uns'rer Jahre). Da die Objekte am Zeilenende stehen und somit den Reim bilden müssen, sind sie in reimenden Paaren abgelegt. Die Wiederholung immer gleicher Reimpaare könnte dadurch ausgeglichen werden, daß drei, vier oder mehr reimende Objekte zur Verfügung gestellt werden, von denen das Programm jeweils zwei zufällig auswählen kann. Die letzte Gruppe bildet eine Sammlung sich reimender Subjekte, die ebenfalls paarweise stehen und, wie schon beschrieben, nach ihrem Genus klassifiziert sind.

Struktur des Gedichtes

Der Titel des Gedichts besteht aus Attribut, Subjekt und Genitivobjekt, das durch Deklination eines Substantivs erzeugt wird.

Die Verse 1 und 3 folgen dem Aufbau: Attribut, Adjektiv mit Endung, Subjekt, Prädikat, erweitertes Objekt. Vers 2 hat die Struktur: Adjektiv, Prädikat, Attribut, Subjekt, und Vers 4: Adjektiv mit Endung, Adjektiv mit Endung, Subjekt. Alle vier Strophen des Gedichts sind nach dem gleichen Schema strukturiert.

Alle diese Bausteine werden im Programmverlauf in entsprechenden Stringvariablen gesammelt und dann zu Titel und Versen zusammengesetzt. Für die Variablen wurden folgende Buchstaben verwendet: **a** (Attribut), **j** (Adjektiv), **e** (Endung), **s** (Subjekt), **p** (Prädikat), **o** (erweitertes Objekt) und **r** (reimgepaartes Subjekt. Daraus ergibt sich als lyrische Struktur:

```
Titel     t$ = a1$ + s1$ + a2$ + s2$ + e$

1. Vers   v1$ = a3$ + j3$ + e3$ + s3$ + p3$ + o3$

2. Vers   v2$ = j4$ + p4$ + a4$ + r4$

3. Vers   v3$ = a5$ + j5$ + e5$ + s5$ + p5$ + o5$

4. Vers   v4$ = j6$ + e6$ + j7$ + e6$ + r6$
```

Das Programm LOVEPOEM.ART sucht die benötigten Bauelemente zufällig und fügt sie in die vorgegebene Struktur ein.

Programm
LOVEPOEM.ART

```
1 REM LOVEPOEM.ART
10 GOSUB 60010
20 DIM a$(2,7),e$(2,7),j$(19),s$(19),p$(
19),o$(9,1),r$(19,1),x$(19)
30 RANDOMIZE TIME
40 GOSUB 10010
1000 s1=INT(RND(1)*20)
1010 x$=s$(s1)::GOSUB 20010:s1$=MID$(s$(
s1),2)
1020 a=INT(RND(1)*8):a1$=a$(g,a)
1030 s2=INT(RND(1)*20):IF s2=s1 GOTO 103
0
1040 x$=s$(s2):x=s2:GOSUB 20010:s2$=MID$
(s$(s2),2)
1050 a2$="des":IF g=0 THEN a2$="der"
1060 e2$="es":IF g=0 THEN e2$=""
1100 t$=a1$+" "+s1$+" "+a2$+" "+s2$+e2$
2000 s3=INT(RND(1)*20)
2010 x$=s$(s3):GOSUB 20010:s3$=MID$(s$(s
3),2)
2020 a=INT(RND(1)*8):a3$=a$(g,a):e3$=e$(
g,a)
2030 j3=INT(RND(1)*20):j3$=j$(j3)
2040 p3=INT(RND(1)*20):p3$=p$(p3)
2050 o3=INT(RND(1)*10):r1=INT(RND(1)*2)
2060 o3$=o$(o3,r1):o5$=o$(o3,(1 XOR r1))
2100 v1$=a3$+" "+j3$+e3$+" "+s3$+" "+p3$
+" "+o3$
3000 j4=INT(RND(1)*20):IF j4=j3 GOTO 300
0
3010 j4$=CHR$(ASC(j$(j4))-32)+MID$(j$(j4
),2)
3020 p4=INT(RND(1)*20):IF p4=p3 GOTO 302
0
3030 p4$=p$(p4)
3040 r4=INT(RND(1)*20):r2=INT(RND(1)*2)
3050 x$=r$(r4,r2):GOSUB 20010:r4$=MID$(r
$(r4,r2),2)
3060 a=INT(RND(1)*8):a4$=CHR$(ASC(a$(g,a
))+32)+MID$(a$(g,a),2)
3070 x$=r$(r4,(1 XOR r2)):GOSUB 20010:r6
$=MID$(r$(r4,(1 XOR r2)),2)
3080 e6$="e":IF g=1 THEN e6$="er"
3090 IF g=2 THEN e6$="es"
3100 v2$=j4$+" "+p4$+" "+a4$+" "+r4$
4000 s5=INT(RND(1)*20):IF s5=s3 GOTO 400
0
4010 x$=s$(s5):GOSUB 20010:s5$=MID$(s$(s
5),2)
4020 a=INT(RND(1)*8):a5$=a$(g,a):e5$=e$(
g,a)
4030 j5=INT(RND(1)*20):IF j5=j4 OR j5=j3
 GOTO 4030
4040 p5=INT(RND(1)*20):IF p5=p4 OR p5=p3
 GOTO 4040
4050 j5$=j$(j5):p5$=p$(p5)
4100 v3$=a5$+" "+j5$+e5$+" "+s5$+" "+p5$
+" "+o5$
5000 j6=INT(RND(1)*20):IF j6=j3 OR j6=j4
 OR j6=j5 GOTO 5000
```

Programm
LOVEPOEM.ART

```
5010 j7=INT(RND(1)*20):IF j7=j3 OR j7=j4
 OR j7=j5 OR j7=j6 GOTO 5010
5020 j6$=CHR$(ASC(j$(j6))-32)+MID$(j$(j6
),2):j7$=j$(j7)
5100 v4$=j6$+e6$+", "+j7$+e6$+" "+r6$
6000 IF z=0 THEN PRINT #8,t$:PRINT #8:PR
INT #8
6010 PRINT #8,v1$
6020 PRINT #8,v2$
6030 PRINT #8,v3$
6040 PRINT #8,v4$
6050 PRINT #8:z=z+1:IF z<4 GOTO 2000
6060 END
10000 REM >Lexikon lesen<
10010 FOR j=0 TO 2:FOR i=0 TO 7:READ a$(
j,i),e$(j,i):NEXT i:NEXT j
10020 FOR j=0 TO 19:READ j$(j):NEXT j
10030 FOR j=0 TO 19:READ s$(j):NEXT j
10040 FOR j=0 TO 19:READ p$(j):NEXT j
10050 FOR j=0 TO 9:FOR i=0 TO 1:READ o$(
j,i):NEXT i:NEXT j
10060 FOR j=0 TO 19:FOR i=0 TO 1:READ r$
(j,i):NEXT i:NEXT j
10070 RETURN
20000 REM >Bestimmung des Genus<
20010 IF LEFT$(x$,1)="f" THEN g=0
20020 IF LEFT$(x$,1)="m" THEN g=1
20030 IF LEFT$(x$,1)="n" THEN g=2
20040 RETURN
50000 REM >Lexikon<
50010 DATA Die,e,Eine,e,Jede,e,Manch',e,
Meine,e,Deine,e,Uns're,e,Eure,e
50020 DATA Der,e,Ein,er,Jeder,e,Manch',e
r,Mein,er,Dein,er,Unser,e,Euer,e
50030 DATA Das,e,Ein,es,Jedes,e,Manch',e
s,Mein,es,Dein,es,Unser,es,Euer,es
50040 DATA schoen,hell,licht,warm,zärtli
ch,golden,befreit,offen,unendlich,duften
d,leuchtend,beschwingt,frei,geloest,erha
ben,befreit,vollendet,rund,frisch,heilig
50050 DATA fSeele,nHerz,fFreude,fLust,fU
nendlichkeit,nGlück,nJauchzen,fErfullung
,mFriede,fSonne,mMond,nMeer,mWind,fBlume
,fStille,fWärme,fUmarmung,mHauch,mGlanz,
nLeben
50060 DATA streichelt,küßt,haucht,hofft,
sehnt,träumt,schwebt,atmet,lebt,pulsiert
,stroemt,fliegt,glüht,hoeht,stillt,beleb
t,verglüht,beruhigt,schmeichelt,erfragt
50070 DATA den Horizont der Stille,den Ü
berfuß der Fülle,das Silberlicht der Ste
rne,das Goettliche der Ferne,das Klopfen
 uns'rer Herzen,das warme Licht von Kerz
en,die Blüte uns'rer Jahre,das Glänzen D
einer Haare
50080 DATA mein Herz in Deiner Hand,was
früher ich nie fand,den Segen stiller St
unden,das Lindern alter Wunden,das Loese
n kleiner Zweiheit,in's Schweben stiller
 Einheit
```

Programm
LOVEPOEM.ART

```
50090 DATA die Tiefe Deiner Augen,den Sa
ft von schwarzen Trauben,die Weichheit D
einer Lippen,die Süße dran zu nippen,die
 Klarheit des Gedankens,das Ende allen W
ankens
50100 DATA fHand,nBand,fLuft,mDuft,nGlüc
k,nStück,fFigur,fNatur,nGesicht,nGedicht
,mTraum,mBaum,fBlüte,fSüße,fGänze,fLenze
,nLachen,nErwachen,fKühle,fGefühle
50110 DATA fFreude,fTreue,mSaft,fKraft,f
Linde,nGebinde,nKüssen,nWissen,fWeide,nG
eschmeide,fKrone,fWonne,nEntzücken,nEntr
ücken,nFinden,nBinden,nBemühen,nErblühen
,nEntdecken,nWecken
60000 REM > Umlaute <
60010 SYMBOL AFTER 91
60020 SYMBOL 91,66,24,60,102,126,102,102
,0
60030 SYMBOL 92,130,56,108,198,198,108,5
6,0
60040 SYMBOL 93,66,0,102,102,102,102,60,
0
60050 SYMBOL 123,68,0,120,12,124,204,118
,0
60060 SYMBOL 124,36,0,60,102,102,102,60,
0
60070 SYMBOL 125,36,0,102,102,102,102,62
,0
60080 SYMBOL 126,60,102,124,102,102,102,
108,96
60090 RETURN
```

Programmbeschreibung
LOVEPOEM.ART

Zeile 10 bedingt einen Sprung zum Unterprogramm ab **Zeile 60010**. Dort werden die ASCII-Zeichen 91 bis 93 und 123 bis 126 durch Punktmuster ersetzt, die auf dem Bildschirm die deutschen Sonderzeichen, Umlaute und »ß«, darstellen. Die Belegung folgt der internationalen Regelung für nationale Sonderzeichen. Wird der angeschlossene Drucker auf den deutschen Zeichensatz eingestellt, druckt er bei Empfang von ASCII 91 ein »Ä« statt des US-ASCII-Zeichens »[«.
Schwierigkeiten ergaben sich, seltsamerweise übrigens nicht bei allen Programmen, durch eine Besonderheit des Schneider CPC. Der ASCII-Code 124 (»¦«) wird verwendet, um auf BASIC-Ebene CP/M-Kommandos zu erteilen, also zum Beispiel eine Datei auf der Diskette zu löschen: ¦ERA, "filename.ext". ASCII 124 ist aber für das deutsche »ö« vorgesehen. Bei der Erstellung dieses Programms ersetzte das System jedes Zeichen nach einem ASCII-124-ö durch ein Leerzeichen. Aus diesem Grund wurde statt »ö« die unschöne Umlautung »oe« verwendet.

Programmbeschreibung
LOVEPOEM.ART

In **Zeile 20** werden die benötigten Stringvariablen dimensioniert, und **Zeile 30** bestimmt mit der Variablen TIME einen Anfang für die Pseudozufallszahlenreihe.
Zeile 40 verzweigt in das Unterprogramm ab **Zeile 10010**, wo die in den **Data-Zeilen 50010 bis 50110** abgelegten Textelemente in die entsprechenden Stringvariablen geladen werden.
Mit **Zeile 1000** beginnt das Programm zu »dichten«. Zuerst wird der Titel des Werkes gesucht. Der Prozeß ist im wesentlichen immer gleich: Eine Zufallszahl wird entsprechend der Menge der zur Auswahl stehenden Elemente ermittelt. Sie wird als Index für das entsprechende Stringarray verwendet. Mit der Zufallszahl s1 wird also der String s1$(s1) als erstes Subjekt bestimmt. Das Subjekt muß immer zuerst gesucht werden, weil sein Genus für die Auswahl des Attributs und der Endung bestimmend ist.
Deshalb verzweigt das Programm nach jeder Subjektwahl in das Unterprogramm ab **Zeile 20010**. Dort wird anhand des ersten Zeichens im Subjektstring (f, m oder n) der Variablen g der Wert 0, 1 oder 2 zugeordnet. In **Zeile 1010** erfolgt das erstmals. Anschließend muß das Genuskennzeichen aus dem String entfernt werden.
In **Zeile 1030** wird das zweite Subjekt für den Titel gesucht, das anschließend in den Genitiv gesetzt wird. Um Wiederholungen wie „Der Friede des Friedens" zu vermeiden, wird überprüft, ob s1 und s2 identisch sind.
Die **Zeilen 1050 und 1060** veranlassen die Deklination. Wenn die Variable g aus dem Unterprogramm ab **Zeile 20010** mit dem Wert 0 zurückgenommen ist, ist das Subjekt feminin. Der Genitivartikel heißt dann »der«; eine Endung hat das feminine Subjekt-Genitiv nicht.
In **Zeile 1100** werden die gefundenen Elemente zum Titel t$ verknüpft.
Die **2000er Zeilen** ermitteln die Elemente für den ersten Vers. Für das Objekt o3$ wird gleich das zugehörige Reimpaar in o5$ erfaßt. Während die Reimpaare in dieser Programmversion zwar unzertrennlich sind, kann doch ihre Reihenfolge gewechselt werden. Die Variable r1 bestimmt als Zufallszahl 0 oder 1 und wählt damit das erste oder zweite Wort des Reimpaares für o3$ aus; o5$ wird nun das andere Wort zugewiesen. Es wird mit 0 aufgerufen, wenn r1 den Wert 1 hatte, mit 1, wenn r1 den Wert 0 hatte. Das ist bedingt durch die Operation: 1 XOR r1 (**Zeile 2060**), die für r1=0 den Wert 1, für r1=1 den Wert 0 ergibt.

Programmbeschreibung LOVEPOEM.ART

Die **3000er Zeilen** stellen den zweiten Vers zusammen. Von nun an wird ständig überprüft, ob ein gefundenes Wort nicht schon verwendet wurde. Unschöne Wiederholungen sollen so ausgeschlossen werden.
Am Anfang des zweiten Verses steht ein Adjektiv. Alle Adjektive sind im Lexikon mit kleinen Anfangsbuchstaben eingetragen. Da jeder Vers mit einem großen Buchstaben beginnen soll, wird in **Zeile 3020** der ASCII-Code des ersten Zeichens des Adjektivs j4$ um den Wert 32 verringert. Die ASCII-Codes der Minuskeln (Kleinbuchstaben) a—z haben einen um 32 größeren Wert als die der Majuskeln (Großbuchstaben) A—Z.
Bei dem Attribut a4$ in **Zeile 3060** ist genau der umgekehrte Vorgang notwendig, da die Attribute im Lexikon mit großen Anfangsbuchstaben eingetragen sind.
In **Zeile 3070** wird nach r4$ auch gleich das Reimpendant r6$ erfaßt und in den **Zeilen 3080 und 3090** die ihm zugehörende Endung e6$ ermittelt.
Die **4000er Zeilen** stellen den dritten Vers zusammen, die **5000er** den vierten.
Die vierte Programmkomponente veranlaßt die Ausgabe des Gedichts. In diesem Fall wurde der Drucker als Medium gewählt. Die Variable z zählt vier Programmdurchläufe, die jeweils eine vierzeilige Strophe erzeugen. Wenn z noch den Wert 0 hat, also nach dem ersten Durchlauf, wird der Titel t$ ausgedruckt und zwei Zeilen Durchschuß gegeben. Die **Zeilen 6010 bis 6040** bewirken den Ausdruck der vier Verse.
Zeile 6050 schießt eine Leerzeile ein und erhöht z um den Wert 1. Solange z kleiner als 4 ist, also noch keine vier Strophen gedruckt wurden, erfolgt ein Rücksprung nach **Zeile 2000**, wo der erste Vers der folgenden Strophe gesucht wird. Sonst endet das Programm mit **Zeile 6060**. Hier könnte aber auch ein Formularvorschub an den Drucker gesendet werden und danach ein Rücksprung nach **Zeile 1000** erfolgen. Das Programm würde dann sofort mit der Zusammenstellung des nächsten Gedichts beginnen. Und so weiter, und so weiter.

Konkurrenz für Poeten?

Handelsübliche Gedichtbände umfassen durchschnittlich hundert Gedichte. Da das Programm selbst in dieser BASIC-Version lediglich 17 Sekunden für die Erstellung der vier Strophen benötigt, kann es in rund 28 Minuten einen kompletten Gedichtband zusammenreimen. Da muß doch jeder arme Poet in seiner lecken Dachkammer blaß vor Neid werden, oder?

Kunst kommt von Ordnung

Im ersten Kapitel wurde die für manchen konservativen Wissenschaftler schwer verdauliche Überlegung angestellt, daß Richtung das kleinste Maß für Intelligenz sei. Nun ist dieses Buch nicht der Ort, einen Theoriestreit über diese Frage mit Argumenten zu schmücken. Unter dem Stichwort Kunst mag aber anklingen, wie durch fortgesetzte Überlagerung von Richtungen immer dichtere Strukturen entstehen.

Wenn ein Künstler vor der leeren Leinwand oder dem weißen Blatt Papier sitzt, wird er meist zuerst das Format durch ein Achsenkreuz in vier Quadranten teilen, um eine Orientierung auf der Fläche zu erhalten. Danach wird das Motiv mit wenigen Strichen skizziert, die nur wesentliche Richtungen angeben. Diese dynamischen Gewichte werden so ausgeglichen, daß die gewünschte Komposition deutlich wird. Nun werden die Richtungsangaben differenziert, bis die Konturen des Motivs im Umriß hervortreten. Die Details werden gestaltet, indem die groben Linien (= Richtungen) soweit durch wechselnde Linien überlagert werden, bis die Vorzeichnung dem gewünschten Bild entspricht. Erst zum Schluß werden die optischen Feinheiten flächig mit den entsprechenden Farbwerten ausgearbeitet.

Was ist »Malen?«

Wahlloses Aufbringen von Farbe auf eine Leinwand, wie es dem surrealistischen Prinzip des automatischen Schreibens entspricht und zum Beispiel von den Vertretern des Action Painting angewendet wurde, kann als künstlerischer Akt theoretisch gedeutet und in bezug zur gesellschaftlichen Realität gesetzt werden. Der Laie bewertet ein Kunstwerk jedoch nach der handwerklichen Fertigkeit seines Urhebers sowie nach der sichtbaren Ordnung, die er in erkennbaren Gegenständen und einer das Bild ordnenden Komposition findet.

Es ist denkbar, daß ein Programm aus einer Sammlung von Formen nach vorgegebenen Kompositionsschemata beliebig Bilder zusammensetzt — ähnlich dem Programm LOVEPOEM.ART, das nach diesem Verfahren Gedichte entstehen läßt. Die Syntax von Bildern unterscheidet sich jedoch von der eines Satzes. So können sich zum Beispiel Gegenstände durch ihre räumliche Anordnung überdecken.

Das Programm KALEIDOS.ART verwendet als Gestaltungselemente einen Satz grafischer Formen, die in ihrer Summe im folgenden Modul genannt werden, weil sie auch gegen einen Satz anderer Elemente ausgetauscht werden können.

Die Elemente des verwendeten Moduls erfüllen folgende Bedingungen: Jedes Element belegt eine Fläche von sechzehn mal sechzehn Bildpunkten (Pixel), die gesetzt (Farbe) oder nicht gesetzt (Hintergrund) sein können. Die Pixel sind so arrangiert, daß zwei beliebige Elemente, die nebeneinander gesetzt werden, immer durchgehende Linien bilden.
Die Bildelemente können also frei kombiniert werden, ohne daß Sprünge oder Absätze entstehen. Sie vereinigen sich zu einer gemeinsamen Gestalt, oder wie es in der Semiotik, der Lehre von den Zeichen, heißen würde, die Zeichen bilden Superzeichen.
Die Komposition der Bildelemente erfolgt wie bei einem Kaleidoskop, das heißt, ein zufällig gesetztes Element wird an vier Achsen gespiegelt: der Mittelsenkrechten, der Mittelwaagerechten und den beiden Diagonalen. Die Bildelemente ordnen sich folglich zu einem punktsymmetrischen Gesamtmuster.

Programmstruktur

Auch dieses Programm besteht aus vier Komponenten. Im ersten Teil werden die Elemente des Moduls gestaltet. Der Schneider CPC bietet die schöne Möglichkeit, die Bitmuster des Zeichensatzes neu zu definieren. Die grafischen Elemente werden also bestimmten ASCII-Codes des Zeichensatzes zugeordnet und können dann als Zeichenstrings aufgerufen werden. Dazu wurden die ASCII-Zeichen 97 (»a«) bis 118 (»w«) verwendet. Jeweils vier umdefinierte ASCII-Zeichen (von 8 x 8 Pixel) setzen sich zu einem Bildelement zusammen.

Jede Schreibstelle auf dem Bildschirm besteht aus 8 x 8 Pixel. Jeweils vier grafisch neu definierte Schriftzeichen werden zu einem Bildelement zusammengesetzt.

Die zweite Komponente wählt per Zufall ein beliebiges Gestaltungselement aus, wobei die Symmetrieregel beachtet werden muß, wenn das Zeichen direkt auf einer der vier Symmetrieachsen liegt.
Die dritte Komponente spiegelt das gefundene Zeichen, und die vierte übermittelt das Ergebnis an den Monitor.

Programm KALEIDOS.ART

```
1 REM KALEIDOS.ART
10 GOSUB 60010
20 BORDER 13:INK 0,13:PAPER 0:CLS
30 WINDOW 9,32,1,24
40 INK 1,14:INK 2,0:PAPER 2:PEN 1:CLS
50 RANDOMIZE TIME
100 x=INT(RND(1)*6)+1:u=2*x
110 y=INT(RND(1)*6)+1:v=2*y
200 IF x=6 AND y=6 THEN GOSUB 4000:GOTO
100
210 IF (x=6 AND y<6) OR (x<6 AND y=6) TH
EN GOSUB 3000:GOTO 100
220 IF x=y AND x<6 THEN GOSUB 2000:GOTO
100
1000 z=INT(RND(1)*27)
1100 IF z<3 THEN FOR j=0 TO 7:z(j)=z:NEX
T j
1130 IF z=3 THEN FOR j=0 TO 6 STEP 2:z(j
)=3:z(j+1)=4:NEXT j
1140 IF z=4 THEN FOR j=0 TO 6 STEP 2:z(j
)=4:z(j+1)=3:NEXT j
1150 IF z=5 THEN z(0)=z:z(1)=7:z(2)=5:z(
3)=8:z(4)=6:z(5)=7:z(6)=6:z(7)=8
1160 IF z=6 THEN z(0)=z:z(1)=8:z(2)=6:z(
3)=7:z(4)=5:z(5)=8:z(6)=5:z(7)=7
1170 IF z=7 THEN z(0)=z:z(1)=5:z(2)=8:z(
3)=5:z(4)=7:z(5)=6:z(6)=8:z(7)=6
1180 IF z=8 THEN z(0)=z:z(1)=6:z(2)=7:z(
3)=6:z(4)=8:z(5)=5:z(6)=7:z(7)=6
1190 IF z=9 THEN z(0)=z:z(1)=z:z(2)=10:z
(3)=10:z(4)=10:z(5)=10:z(6)=z:z(7)=z
1200 IF z=10 THEN z(0)=z:z(1)=z:z(2)=9:z
(3)=9:z(4)=9:z(5)=9:z(6)=z:z(7)=z
1210 IF z=11 THEN z(0)=z:z(1)=z:z(2)=12:
z(3)=12:z(4)=14:z(5)=14:z(6)=13:z(7)=13
1220 IF z=12 THEN z(0)=z:z(1)=14:z(2)=11
:z(3)=13:z(4)=13:z(5)=11:z(6)=14:z(7)=12
1230 IF z=13 THEN z(0)=z:z(1)=z:z(2)=14:
z(3)=14:z(4)=12:z(5)=12:z(6)=11:z(7)=11
1240 IF z=14 THEN z(0)=z:z(1)=12:z(2)=13
:z(3)=11:z(4)=11:z(5)=13:z(6)=12:z(7)=14
1250 IF z=15 THEN z(0)=z:z(1)=z:z(2)=16:
z(3)=16:z(4)=18:z(5)=18:z(6)=17:z(7)=17
1260 IF z=16 THEN z(0)=z:z(1)=18:z(2)=15
:z(3)=17:z(4)=17:z(5)=15:z(6)=18:z(7)=16
1270 IF z=17 THEN z(0)=z:z(1)=z:z(2)=18:
z(3)=18:z(4)=16:z(5)=16:z(6)=15:z(7)=15
1280 IF z=18 THEN z(0)=z:z(1)=16:z(2)=17
:z(3)=15:z(4)=15:z(5)=17:z(6)=16:z(7)=18
```

Programm
KALEIDOS.ART

```
1290 IF z>18 THEN FOR j=0 TO 7:z(j)=19:N
EXT j
1300 d=7:GOSUB 10010
1310 LOCATE u,v:PRINT z$(0,0):LOCATE u,v
+1:PRINT z$(0,1)
1320 LOCATE v,u:PRINT z$(1,0):LOCATE v,u
+1:PRINT z$(1,1)
1330 LOCATE u,24-v:PRINT z$(2,0):LOCATE
u,25-v:PRINT z$(2,1)
1340 LOCATE v,24-u:PRINT z$(3,0):LOCATE
v,25-u:PRINT z$(3,1)
1350 LOCATE 24-u,v:PRINT z$(4,0):LOCATE
24-u,v+1:PRINT z$(4,1)
1360 LOCATE 24-v,u:PRINT z$(5,0):LOCATE
24-v,u+1:PRINT z$(5,1)
1370 LOCATE 24-u,24-v:PRINT z$(6,0):LOCA
TE 24-u,25-v:PRINT z$(6,1)
1380 LOCATE 24-v,24-u:PRINT z$(7,0):LOCA
TE 24-v,25-u:PRINT z$(7,1)
1390 GOTO 100
2000 z=INT(RND(1)*13)
2010 IF z<3 THEN FOR j=0 TO 3:z(j)=z:NEX
T j
2020 IF z=3 THEN z(0)=9:z(1)=10:z(2)=9:z
(3)=10
2030 IF z=4 THEN z(0)=10:z(1)=9:z(2)=10:
z(3)=9
2040 IF z=5 THEN z(0)=11:z(1)=12:z(2)=13
:z(3)=14
2050 IF z=6 THEN z(0)=13:z(1)=14:z(2)=11
:z(3)=12
2060 IF z=7 THEN z(0)=15:z(1)=16:z(2)=17
:z(3)=18
2070 IF z=8 THEN z(0)=17:z(1)=18:z(2)=15
:z(3)=16
2080 IF z>8 THEN FOR j=0 TO 7:z(j)=19:NE
XT j
2100 d=3:GOSUB 10010
2200 LOCATE u,u:PRINT z$(0,0):LOCATE u,u
+1:PRINT z$(0,1)
2210 LOCATE u,24-u:PRINT z$(1,0):LOCATE
u,25-u:PRINT z$(1,1)
2220 LOCATE 24-u,24-u:PRINT z$(2,0):LOCA
TE 24-u,25-u:PRINT z$(2,1)
2230 LOCATE 24-u,u:PRINT z$(3,0):LOCATE
24-u,u+1:PRINT z$(3,1)
2240 RETURN
3000 z=INT(RND(1)*11)
3010 IF z<3 THEN FOR j=0 TO 3:z(j)=z:NEX
T j
3020 IF z=3 THEN z(0)=z:z(1)=4:z(2)=z:z(
3)=4
3030 IF z=4 THEN z(0)=z:z(1)=3:z(2)=z:z(
3)=3
3040 IF z=5 THEN z(0)=7:z(1)=5:z(2)=8:z(
3)=6
3050 IF z=6 THEN z(0)=8:z(1)=6:z(2)=7:z(
3)=5
3060 IF z>6 THEN FOR j=0 TO 7:z(j)=19:NE
XT j
```

Programm
KALEIDOS.ART

```
3100 d=3:GOSUB 10010
3110 IF u<12 THEN v=u:u=12
3200 LOCATE u,v:PRINT z$(0,0):LOCATE u,v
+1:PRINT z$(0,1)
3210 LOCATE v,u:PRINT z$(1,0):LOCATE v,u
+1:PRINT z$(1,1)
3220 LOCATE 24-u,24-v:PRINT z$(2,0):LOCA
TE 24-u,25-v:PRINT z$(2,1)
3230 LOCATE 24-v,24-u:PRINT z$(3,0):LOCA
TE 24-v,25-u:PRINT z$(3,1)
3240 RETURN
4000 z=INT(RND(1)*4)
4010 z(0)=z:IF z=3 THEN z(0)=19
4100 d=0:GOSUB 10010
4200 LOCATE 12,12:PRINT z$(0,0):LOCATE 1
2,13:PRINT z$(0,1)
4210 RETURN
10000 REM >Kennzahl-Zeichen-Umwandlung<
10010 FOR j=0 TO d
10020 IF z(j)=0 THEN z$(j,0)="ab":z$(j,1
)="cd"
10030 IF z(j)=1 THEN z$(j,0)="ef":z$(j,1
)="gh"
10040 IF z(j)=2 THEN z$(j,0)="ij":z$(j,1
)="kl"
10050 IF z(j)=3 THEN z$(j,0)="mn":z$(j,1
)="op"
10060 IF z(j)=4 THEN z$(j,0)="qr":z$(j,1
)="st"
10070 IF z(j)=5 THEN z$(j,0)="eb":z$(j,1
)="gd"
10080 IF z(j)=6 THEN z$(j,0)="af":z$(j,1
)="ch"
10090 IF z(j)=7 THEN z$(j,0)="ef":z$(j,1
)="cd"
10100 IF z(j)=8 THEN z$(j,0)="ab":z$(j,1
)="gh"
10110 IF z(j)=9 THEN z$(j,0)="eb":z$(j,1
)="ch"
10120 IF z(j)=10 THEN z$(j,0)="af":z$(j,
1)="gd"
10130 IF z(j)=11 THEN z$(j,0)="eb":z$(j,
1)="cd"
10140 IF z(j)=12 THEN z$(j,0)="ab":z$(j,
1)="gd"
10150 IF z(j)=13 THEN z$(j,0)="ab":z$(j,
1)="ch"
10160 IF z(j)=14 THEN z$(j,0)="af":z$(j,
1)="cd"
10170 IF z(j)=15 THEN z$(j,0)="ef":z$(j,
1)="gd"
10180 IF z(j)=16 THEN z$(j,0)="eb":z$(j,
1)="gh"
10190 IF z(j)=17 THEN z$(j,0)="af":z$(j,
1)="gh"
10200 IF z(j)=18 THEN z$(j,0)="ef":z$(j,
1)="ch"
10210 IF z(j)=19 THEN z$(j,0)="uv":z$(j,
1)="vu"
10220 NEXT j
```

Programm
KALEIDOS.ART

```
10230 RETURN
60000 REM >Subzeichendefinition<
60010 SYMBOL AFTER 97
60020 SYMBOL 97,&3,&3,&5,&5,&9,&30,&C0,&
F8
60030 SYMBOL 98,&C0,&C0,&A0,&A0,&90,&C,&
3,&1F
60040 SYMBOL 99,&F8,&C0,&30,&9,&5,&5,&3,
&3
60050 SYMBOL 100,&1F,&3,&C,&90,&A0,&A0,&
C0,&C0
60060 SYMBOL 101,&3,&3,&7,&7,&F,&3F,&FF,
&FE
60070 SYMBOL 102,&C0,&C0,&E0,&E0,&F0,&FC
,&FF,&7F
60080 SYMBOL 103,&FE,&FF,&3F,&F,&7,&7,&3
,&3
60090 SYMBOL 104,&7F,&FF,&FC,&F0,&E0,&E0
,&C0,&C0
60100 SYMBOL 105,&3,&3,&3,&3,&3,&3,&FF,&
FE
60110 SYMBOL 106,&C0,&C0,&C0,&C0,&C0,&C0
,&FF,&7F
60120 SYMBOL 107,&FE,&FF,&3,&3,&3,&3,&3,
&3
60130 SYMBOL 108,&7F,&FF,&C0,&C0,&C0,&C0
,&C0,&C0
60140 SYMBOL 109,&3,&3,&3,&3,&3,&3,&C3,&
FB
60150 SYMBOL 110,&C0,&C0,&C0,&C0,&C0,&C0
,&C3,&DF
60160 SYMBOL 111,&FB,&C3,&3,&3,&3,&3,&3,
&3
60170 SYMBOL 112,&DF,&C3,&C0,&C0,&C0,&C0
,&C0,&C0
60180 SYMBOL 113,&3,&3,&1,&1,&1,&0,&FF,&
FF
60190 SYMBOL 114,&C0,&C0,&80,&80,&80,&0,
&FF,&FF
60200 SYMBOL 115,&FF,&FF,&0,&1,&1,&1,&3,
&3
60210 SYMBOL 116,&FF,&FF,&0,&80,&80,&80,
&C0,&C0
60220 SYMBOL 117,&3,&1,&0,&0,&0,&0,&80,&
C0
60230 SYMBOL 118,&C0,&80,&0,&0,&0,&0,&1,
&3
60240 RETURN
```

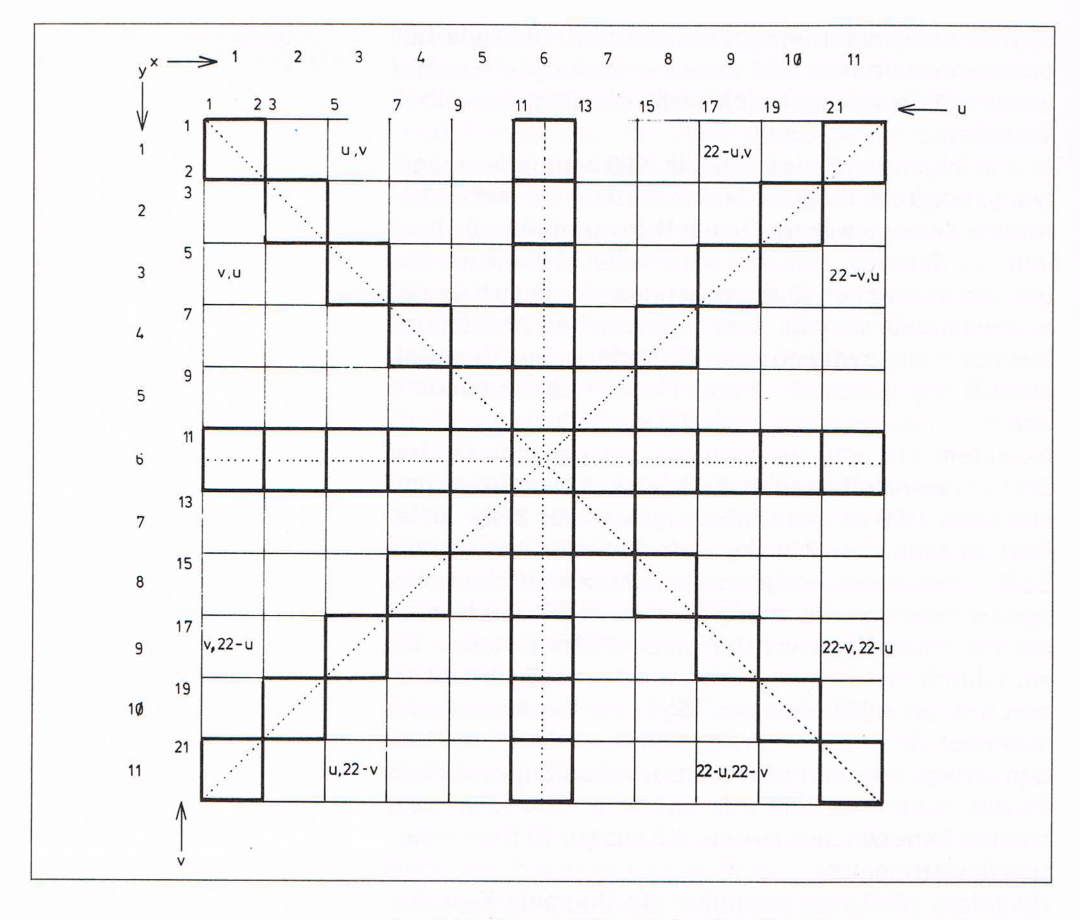
Programmbeschreibung
KALEIDOS.ART

Zeile 10 bedingt eine Verzweigung ins Unterprogramm ab **Zeile 60010**, wo den ASCII-Zeichen »a« bis »w« die Pixeldaten zugeordnet werden. Die Umdefinition der ASCII-Zeichen ist übrigens leider nur durch Zurücksetzen des Computers widerrufbar.
In **Zeile 20** werden die verwendeten Farbwerte und in **Zeile 30** wird ein Fenster definiert. Dies erleichtert die Umrechnung von Bildpositionen auf Bildschirmkoordi-

naten, weil die Position 1,1 nicht mehr der linken oberen Ecke des Monitors, sondern der linken oberen Ecke der zu gestaltenden Bildfläche entspricht. Zur einfachen Anwendung der Symmetrieregel ist das Gestaltungsformat quadratisch gewählt.
Die **Zeilen 100 und 110** finden Zufallswerte für x und y, aus denen Positionen (u,v) im Bildschirmfenster errechnet werden.
Haben x und y beide den Wert 6, ergibt sich daraus die Bildposition genau im Zentrum. Um der Symmetrieregel zu genügen, darf nur ein in sich punktsymmetrisches Bildelement hierher gesetzt werden. Aus diesem Grund veranlaßt **Zeile 200** dann einen Sprung nach **Zeile 4000**.
Auf einer der Mittelsenkrechten liegt die gefundene Position, wenn entweder x oder y den Wert 6 hat und der zweite Wert von 6 verschieden ist. Da die dann zu wählenden Bildelemente achsensymmetrisch zur Senkrechten oder Waagerechten sein müssen, wird in **Zeile 210** nach **Zeile 3000** verzweigt, wenn diese Bedingung erfüllt ist.
Wenn x und y denselben Wert haben, liegt die gefundene Position auf einer Diagonalen, und es müssen entsprechende Elemente ausgewählt werden. Die Bedingung in **Zeile 220**, daß der x=y-Wert auch noch kleiner 6 sein soll, ist eigentlich überflüssig. Gilt x=y=6, gelangt das Programm nämlich gar nicht in diese Zeile, weil es bereits in **Zeile 200** nach **Zeile 4000** verzweigt wird.
Zeile 1000: Trifft keine der genannten Bedingungen zu, wird eine Position in den Bereichen zwischen den Symmetrieachsen gefunden. Hier kann jedes beliebige Bildelement eingesetzt werden.
Das Modul besteht aus zwanzig (0 bis 19) Superzeichen, von denen die Nummer 19 fast leer ist. Sie dient der optischen Auflockerung des Ornaments, das nach und nach auf dem Monitor wächst und sich ständig verändert. Die Zufallszahl z bestimmt als Kennzahl das Superzeichen. In **Zeile 1000** wird für z ein Zufallswert von 0 bis 26 ermittelt. Mit allen Werten größer 18 wird das optisch auflockernde Leerzeichen aufgerufen, das also statistisch häufiger verwendet wird als die übrigen Superzeichen. Die Superzeichen 0 bis 2 sind in sich punktsymmetrisch gestaltet. Durch Spiegelung an einer der vier Achsen ergibt sich also immer wieder dasselbe Zeichen in der gleichen Ausrichtung auf der Bildfläche. Das heißt, das Zeichen kann in alle sieben Positionen, die durch Spiegelung an den vier Achsen mit der eingangs ermittelten Position x,y korrespondieren, geschrieben werden, oh-

Programmbeschreibung
KALEIDOS.ART

ne daß die Symmetrieregel verletzt wird. Die **Zeile 1100** ordnet deshalb den acht Superzeichen z(j), die später auf den Bildschirm gebracht werden sollen, denselben Wert z zu.
In den folgenden **Zeilen 1130 bis 1290** wird jedes mögliche z abgefragt. Die jeweils symmetrisch korrespondierenden Zeichen werden in den Feldvariablen z(j) abgelegt. Die Superzeichen sind so gestaltet, daß die jeweiligen symmetrischen Entsprechungen ebenfalls Elemente des Moduls sind. Zu dem Superzeichen 3 bildet das Zeichen 4 die symmetrische Ergänzung; aus dem Zeichen 11 leiten sich 12, 13 und 14 durch Spiegelung ab usw.
Nachdem die acht symmetrisch korrespondierenden Zeichen ermittelt worden sind, springt das Programm aus **Zeile 1300** in das Unterprogramm ab **Zeile 10010**. Dort werden die acht Superzeichen z(0) bis z(7) aus 2 x 2 elementaren Zeichen zusammengesetzt, deren Bitmuster eingangs den umdefinierten ASCII-Zeichen »a« bis »w« zugeordnet wurden. Jedes Superzeichen z(j) wird durch zwei Strings ersetzt, von denen z$(j,0) die beiden oberen ASCII-Zeichen, z$(j,1) die beiden unteren aufnimmt. So werden aus dem Superzeichen 0 die beiden Strings »ab« und »cd« oder aus dem Superzeichen 18 die beiden Strings »ef« und »ch«. Die insgesamt zwanzig Superzeichen setzen sich aus nur 23 Elementarzeichen zusammen.
Nachdem das Unterprogramm aus den acht Kennzahlen z(0) bis z(7) die entsprechenden Elementarzeichen-Strings ermittelt hat, erfolgt ab **Zeile 1310** die Ausgabe auf dem Bildschirm. Dazu müssen aus der einen, anfangs zufällig ermittelten Position x,y die acht symmetrischen Positionen errechnet werden. Danach können die entsprechenden, in der Feldvariablen z$(7,1) abgelegten Elementarzeichen-Strings ausgegeben werden.
Die Gestaltungsfläche ist 11 x 11 Superzeichen groß. Jedes Superzeichen setzt sich aus 2 x 2 Elementarzeichen zusammen. Auf dem Monitor werden also 22 x 22 Schreibstellen (mode 1) bearbeitet. Deshalb wurden bereits in den **Zeilen 100 und 110** die Werte x und y mit 2 multipliziert, um die Bildschirmposition (u,v) zu ermitteln.
Aus den Werten der Bildschirmposition (u,v) werden die Werte für die übrigen sieben symmetrischen Bildschirmpositionen errechnet. Durch die Spiegelung an der Diagonalen von links oben nach rechts unten ergibt sich aus (u,v) die Bildschirmposition (v,u); durch Spiegelung an der Mittelsenkrechten (22-u,v) oder durch Spiegelung an der anderen Diagonalen (22-v,22-u) usw.



Eine Bildschirmposition mit den Koordinaten (u,v), bezogen auf die linke obere Ecke, wird an den orthogonalen und diagonalen Symmetrieachsen gespiegelt.

Programmbeschreibung
KALEIDOS.ART

Die **Zeilen 1310 bis 1380** schreiben die zugehörigen Elementarzeichen-Strings in die passenden Bildschirmpositionen. In **Zeile 1390** wird das Programm nach **Zeile 100** zurückgeführt, wo ein neuer Durchlauf beginnt.
Wurde in den **Zeilen 100 und 110** eine Bildposition auf einer der Diagonalen gefunden, stehen nur die neun zur Diagonalen symmetrischen Superzeichen zur Verfügung, und es sind auch nur vier Bildpositionen zu bearbeiten. Der **Programmblock 2000 bis 2240** ist deshalb bei gleicher Vorgehensweise entsprechend kürzer.
Ebenso verhält es sich mit dem **Programmblock 3000 bis 3240**, der abgearbeitet wird, wenn eine Bildposition auf einer der Mittelsenkrechten gefunden wurde.
Die **Zeilen 4000 bis 4210** werden bearbeitet, wenn ein Element in das Zentrum geschrieben werden soll. Das Modul hält vier punktsymmetrische Superzeichen bereit.

Intelligentes BASIC?

Müssen wir schon wieder umdenken? Es ist noch gar nicht so lange her, da wurden Computer halb witzelnd, aber auch etwas ehrfürchtig-ängstlich Elektronengehirne genannt. Dabei waren die Maschinen jener Zeit kaum leistungsfähiger als moderne Taschenrechner.
Der Begriff Elektronengehirn ist inzwischen aus der Mode gekommen. Wir haben uns an das Fremdartige gewöhnt und erleichtert festgestellt, daß Computer an sich dumm sind. Erst ein Programm erweckt sie zu ihrem elektronischen Leben. Ein Programm, das ein Mensch schreiben muß. Die Ehrfurcht hat sich auf die Programmierer verlagert, die manchmal wie moderne Alchemisten bestaunt werden. Und jetzt sollen Computer doch intelligent sein?
Natürlich nicht. Es sind immer noch die menschlichen Konstrukteure und Programmierer, die Systeme entwerfen. Doch hat der rasante Fortschritt dieser Technologie eine Stufe erreicht, auf der sich die Möglichkeit erschließt, hochkomplexe Aufgaben elektronisch zu bewältigen.
Während die Computer durch fortschreitende Vereinfachung der Konstruktion und der Produktion immer billiger und trotzdem noch leistungsfähiger werden, erfordert die Entwicklung entsprechender Programme immer größere Anstrengungen, immer mehr Mann-Stunden, die sich im Preis niederschlagen. In großen Softwarehäusern überlegt man bereits, wann die Hardware so billig sein wird, daß sie als kostenlose Zugabe in den Preis für eine Software einkalkuliert werden kann. Vielleicht gibt es eines Tages die Trennung von Hard- und Software nicht mehr. Der Vorteil bestünde darin, daß die Technik der jeweiligen Computer vollkommen auf die Notwendigkeiten der Software abgestellt werden kann. Für Anwendungen im KI-Bereich werden bereits Spezialcomputer gebaut.

Schon bald: »Hardware im Preis inbegriffen«?

Theoretisch können aber intelligente Programme für jeden Rechner geschrieben werden, denn alle Computer sprechen die gleiche Sprache. Ihr Wortschatz besteht aus ganzen zwei Zuständen: Spannung (ein) und Nicht-Spannung (aus), was in Zahlen ausgedrückt 0 und 1 geschrieben wird.

Diese Sprache ist die einfachste denkbare überhaupt. Für den menschlichen Geist ist sie allerdings so unhandlich, daß schon früh Übersetzer (Interpreter) entwickelt wurden, die auch Programmier- oder Computersprachen genannt werden. Interpreter sind in der Regel auf spezielle Anwendungsgebiete abgestimmt. Frühe Entwicklungen wie ALGOL, COBOL, FORTRAN waren für wissenschaftliche, technische bzw. wirtschaftliche Aufgabenstellungen bestimmt. Die weite Verbreitung von Heimcomputern war möglich, weil es eine relativ simple Allzwecksprache gab: BASIC.

Jeder Interpreter besteht aus einer Menge von Kommandos und Funktionen, die zu Anweisungen aneinandergereiht werden. Dabei sind, wie auch in der Grammatik, bestimmte Regeln (Syntax) zu beachten. Der Begriff Computersprache ist also angemessen. Der Interpreter übersetzt die in der höheren Sprache abgefaßten Programmanweisungen in die Maschinensprache. Er erleichtert also das Schreiben von Programmen, weil die verwendete Programmiersprache unserer Alltagssprache und unserem Denken verwandter ist als eine Kette von Hunderttausenden von Nullen und Einsern.

BASIC ähnelt einer kindlich-einfachen Umgangssprache — nur in Englisch

BASIC ist deshalb so einfach zu erlernen, weil es sich fast um eine kindlich-einfache Form der englischen Umgangssprache handelt. Die Schwierigkeit des Programmierens reduziert sich darauf, die Aufgabenstellung logisch zu strukturieren und mit dem beschränkten verfügbaren Vokabular auszudrücken.

Der Nachteil jedes Interpreters ist darin zu sehen, daß die Aufgabe zuerst vom Programmierer in die höhere Programmiersprache übersetzt und dann vom Interpreter in die Maschinensprache übertragen werden muß. Da jede Programmiersprache nur begrenzte Ausdrucksmöglichkeiten bereit hält, kann nicht immer die einfachste mögliche Form eines Programms an die Maschine übergeben werden. Die Folge ist, daß mehr Speicherplatz und längere Bearbeitungszeiten benötigt werden. Ein BASIC-Programm läuft, je nach Aufgabenstellung, zehn- bis tausendmal langsamer ab als ein Maschinenspracheprogramm, das die gleiche Arbeit leistet.

Intelligente Programme sind theoretisch also keine Frage der Programmiersprache. Nur die Bewältigung bestimmter Aufgaben ist mit der Allzwecksprache BASIC unpraktikabel. So wird die Struktur der BASIC-Programme schnell verworren, weil es einer besonderen Anstrengung bedarf, sozusagen mit Urlauten intelligente Aussagen zu machen.

BASIC verfügt über ganze drei Vokabeln, die als intelligent bezeichnet werden können, weil sie nicht nur eine Regel in Form einer Bedingung aufnehmen, sondern auch das Erfülltsein oder Nicht-Erfülltsein dieser Bedingung überwachen und entsprechend reagieren.
IF-THEN (-ELSE) trifft intelligente Entscheidungen nach dem Muster: wenn Regen, dann Regenschirm (sonst Sonnenschirm); IF Bedingung wahr THEN Anweisung. Enthält der verwendete BASIC-Dialekt die Erweiterung ELSE, kann eine zusätzliche Anweisung vorgegeben werden, die ausgeführt wird, wenn die IF-Bedingung nicht zutrifft. IF-Bedingungen müssen immer in Form einer Vergleichsoperation vorgegeben werden, zum Beispiel: A<5; B>X; A OR B; (A=B) AND (B<>C) usw., oder sie werden in einem String definiert, zum Beispiel: T$="Regen". Die Befehlszeile kann dann lauten: IF T$="Regen" THEN PRINT "Regenschirm" ELSE PRINT "Sonnenschirm".
Der »Intelligenzquotient« der beiden anderen angesprochenen BASIC-Vokabeln ist noch etwas geringer. **FOR-NEXT** ist das Arbeitstier, das die Bearbeitung einer bestimmten Anweisung oder eines ganzen Programmblocks beliebig oft wiederholt und selbst überwacht, wann das Maß voll ist.
Mit der Anweisung FOR X=A TO B wird die Routinearbeit aufgenommen. Die Variable X dient als Schleifenzähler. Sie wird durch das FOR-Kommando auf den Anfangswert A gesetzt. Dann werden die folgenden Programmanweisungen bearbeitet. Findet BASIC irgendwann ein NEXT X, ist das Ende des Programmteils erreicht, der mehrfach bearbeitet werden soll. Nun wird überprüft, ob der Schleifenzähler X schon den mit B bestimmten Endwert erreicht hat. Trifft dies nicht zu, setzt das BASIC-Programm den Wert von X um 1 herauf und beginnt bei der Anweisung direkt hinter dem FOR-Kommando mit dem nächsten Durchlauf.
Mit der Erweiterung STEP kann der Schleifenzähler auch rückwärts zählen (STEP −1). Die Schrittweite ist frei bestimmbar (STEP 142 oder STEP .053). Erreicht BASIC das NEXT X, und der Wert von X ist ≧B, wird kein Rücksprung mehr ausgelöst.
Der dritte Schlaumeier heißt **ON**. Er fügt sich in die Formulierung: ON (Bedingung) (Sprung-Anweisung). Nicht alle BASIC-Dialekte enthalten dieses Kommando, und es können auch nur bestimmte Bedingungen vorgegeben werden. Am gebräuchlichsten ist die Weichensteller-Bedingung: ON X GOTO (Zeilennummer) oder GOSUB (Zeilennummer). Für diesen bedingten Sprung können mehrere Zeilennummern vorgegeben werden.

Sie werden, durch Kommata voneinander getrennt, hinter die Anweisung GOTO oder GOSUB geschrieben, zum Beispiel: ON X GOTO 100,120,2100,30 etc. Der absolute Wert von X bestimmt das Sprungziel. Ist X zum Beispiel 3, wird die dritte hinter GOTO angegebene Zeilennummer angesprungen. Variationen der ON-Weiche verwenden andere Bedingungen als Sprungauslöser: ON ERROR, ON INTERVAL etc.
Mit diesen drei Hilfsassistenten alleingelassen, sollen wir nun die ganze Brillanz unseres messerscharfen Intellekts in den Rechner füllen! Da sollten wir uns wohl erst einmal ein kleines Zwischenspiel gönnen.

Warum KI-Forscher Spiele mögen

Es gibt Anzeichen dafür, daß intelligente Leute Spiele mögen. Zum Beispiel beschäftigen sich KI-Forscher gerne mit Spielen. Das hat natürlich seine Gründe. Das Spiel ist im Gegensatz zur Realität in Raum, Zeit und Regeln klar umrissen und deshalb viel leichter zu fassen. Man denkt sich einen Spielraum aus und formuliert ein Ziel, mit dessen Erreichen das Spiel definitiv beendet ist. Die möglichen Handlungen für den oder die Spieler werden auf erlaubte Züge begrenzt und in (Spiel-) Regeln gefaßt.
Das kleine Zwischen-Spiel, das jetzt beschrieben wird, ist ein Beispiel dafür, daß wir Entscheidungen viel diffuser treffen, als unser Intellekt eingestehen mag. Schon einfache Situationen können komplexe Konflikte auslösen.
Das Spiel ist unter dem Namen »Goldene Zehn« bekannt. Der Zufall des Würfels trifft beide Spieler gleich. Nur wer die wirkungsvollere Strategie verfolgt, wird das bessere Ergebnis erwirtschaften. Um diesen Streit auszutragen, werden drei Würfel und für beide Spieler Papier und Bleistift hinter einem Sichtschutz benötigt. Jeder schreibt die Zahlen von 1 bis 10 untereinander auf einen Zettel und setzt jeweils ein Multiplikationszeichen dahinter. Die folgenden zehn Würfe sollen nach und nach in beliebiger Reihenfolge diesen zehn Multiplikatoren so zugeordnet werden, daß alle Produkte zusammen am Ende eine möglichst hohe Summe ergeben.
Es wird gewürfelt. Die Augen der drei Würfel werden addiert. Die so ermittelte Zahl trägt jeder Mitspieler in eine Zeile seiner Tabelle ein. Der Clou verbirgt sich darin, daß eine 12 in Zeile 5 eingetragen 60 Punkte bringt, in Zeile 8 jedoch 96. Fällt eine 15, soll man sie in Zeile 10 eintragen oder auf einen noch höheren Wurf warten, der dann vielleicht gar nicht mehr kommt?
Nach zehn Würfen endet die Partie. Jeder multipliziert die Werte der einzelnen Zeilen und addiert sie zum Gesamtergebnis. Wer die höhere Zahl erreicht, hat ohne

Zweifel die besseren Entscheidungen getroffen. Das Glück bleibt bei diesem Spiel trotz der Würfel außen vor. Beide Kontrahenten haben exakt die gleichen Chancen.
Trotzdem, deshalb: Entscheiden Sie sich spontan, stellen Sie also nicht erst Berechnungen im Kopf, mit dem Taschenrechner oder sonstwo an, werden Sie nur mit viel Glück gegen einen Computer gewinnen können. Probieren Sie's aus.

Programm GOLD10. GEM

```
1 REM GOLD10.GEM
10 GOSUB 10000
20 z=z+1
30 PRINT"      Bitte drücken Sie eine Tas
te."
40 PRINT"                um zu würfeln."
50 GOSUB 11000
100 CLS
110 PEN 1:PRINT"             Wo wollen Sie de
n Wert einordnen?"
120 INPUT"Zahl von 1 bis 10 und <RETURN>
 eingeben";i
130 IF i<1 THEN CLS:PEN 3:PRINT,"
   Unzulässige Eingabe!":INPUT"Bitte neu
 eigeben:";i:GOTO 130
140 IF i>10 THEN CLS:PEN 3:PRINT,"
    Unzulässige Eingabe!":INPUT"Bitte ne
u eigeben:";i:GOTO 130
150 IF u(i-1)<>0 THEN CLS:PEN 3:PRINT,"
   Die Position ist bereits belegt!":INP
UT"Bitte neu eingeben:";i:GOTO 130
160 u(i-1)=w:uu=i*w
170 LOCATE #2,5,3+i:PRINT #2,USING"##";w
180 LOCATE #2,8,3+i:PRINT #2,USING"###";
uu
200 FOR j=0 TO 9:x(j)=INT(ABS(w-(3+(5/3)
*j))):NEXT j:GOSUB 13000
210 cc=(p+1)*w
220 LOCATE #3,5,4+p:PRINT #3,USING"##";w
230 LOCATE #3,8,4+p:PRINT #3,USING"###";
cc
300 IF z<10 GOTO 20
310 CLS:PRINT"                      ENDE":FO
R w=0 TO 500:NEXT w
320 FOR j=0 TO 9:us=us+u(j)*(j+1):cs=cs+
c(j)*(j+1):NEXT j:e=us-cs:ea=SGN(e):eb=A
BS(e)
330 CLS:PRINT"Summe Anwender";us;"  Summ
e Computer";cs;
340 IF ea=-1 THEN PRINT"Der Computer gew
innt mit";eb;" Punkten";
350 IF ea=1 THEN PRINT"Der Anwender gewi
nnt mit";eb;" Punkten";
360 IF ea=0 THEN PRINT"        U N E N T
S C H I E D E N";
370 t$=""
380 t$=INKEY$
```

Programm GOLD10. GEM

```
390 IF t$="" GOTO 380
400 CLEAR:GOTO 10
10000 REM > UP Bildschirmgestaltung <
10010 INK 0,9:INK 1,26:INK 2,0:INK 3,6
10020 BORDER 26:PAPER 1:CLS
10030 WINDOW #0,1,40,8,25:WINDOW #1,9,31
,1,7:WINDOW #4,1,8,1,7:WINDOW #5,32,40,1
,7
10040 PAPER #1,1:PAPER #4,1:PAPER #5,1
10050 CLS #0:CLS #1:CLS #4:CLS #5
10060 MOVE 130,396:DRAW 492,396,0:DRAW 4
92,290:DRAW 130,290:DRAW 130,396
10100 DEG:y=368
10110 FOR j=0 TO 2
10120 READ x:MOVE x,y+8
10130 FOR w=1 TO 90:DRAW x-8*SIN(w),y+8*
COS(w),0:NEXT w
10140 DRAW x-8,y-48
10150 FOR w=91 TO 180:DRAW x-8*SIN(w),y-
48+8*COS(w):NEXT w
10160 DRAW x+48,y-56
10170 FOR w=181 TO 270:DRAW x+48-8*SIN(w
),y-48+8*COS(w):NEXT w
10180 DRAW x+56,y
10190 FOR w=271 TO 360:DRAW x+48-8*SIN(w
),y+8*COS(w):NEXT w
10200 DRAW x,y+8
10210 NEXT j
10220 MOVE x-16,y:FILL 0
10300 BORDER 9:PAPER #0,0:PAPER #4,0:PAP
ER #5,0
10310 CLS #0:CLS #4:CLS #5
10320 RESTORE
10330 FOR j=0 TO 2
10340 READ x:MOVE x,y-56
10350 DRAW x+48,y-56,2
10360 FOR w=181 TO 270:DRAW x+48-8*SIN(w
),y-48+8*COS(w):NEXT w
10370 DRAW x+56,y
10380 NEXT j
10400 SYMBOL AFTER 91
10410 SYMBOL 91,66,24,60,102,126,102,102
,0
10420 SYMBOL 92,130,56,108,198,198,108,5
6,0
10430 SYMBOL 93,66,0,102,102,102,102,60,
0
10440 SYMBOL 123,68,0,120,12,124,204,118
,0
10450 SYMBOL 124,36,0,60,102,102,102,60,
0
10460 SYMBOL 125,36,0,102,102,102,102,62
,0
10470 SYMBOL 126,60,102,124,102,102,102,
108,96
10480 SYMBOL 202,0,60,126,126,122,118,60
,0
10500 FOR j=3 TO 5:PEN #1,2
10510 LOCATE #1,4,j:PRINT #1,CHR$(202)
10520 LOCATE #1,6,j:PRINT #1,CHR$(202)
```

Programm
GOLD10.GEM

```
10530 LOCATE #1,11,j:PRINT #1,CHR$(202)
10540 LOCATE #1,13,j:PRINT #1,CHR$(202)
10550 LOCATE #1,18,j:PRINT #1,CHR$(202)
10560 LOCATE #1,20,j:PRINT #1,CHR$(202)
10570 NEXT j
10600 BORDER 0:PAPER #0,2:PAPER #4,2:PAP
ER #5,2
10610 CLS #0:CLS #4:CLS #5
10620 PAPER #0,2:PEN #0,3:WINDOW #0,1,40
,9,10
10700 WINDOW #2,9,19,12,25:PAPER #2,0:CL
S #2
10710 WINDOW #3,21,31,12,25:PAPER #3,0:C
LS #3
10720 PEN #2,1:PEN #3,1
10730 LOCATE #2,2,2:PRINT #2,">Anwender"
10740 LOCATE #3,2,2:PRINT #3,"Computer<"
10750 PEN #2,2:PEN #3,2
10760 FOR j=1 TO 9
10770 LOCATE #2,2,3+j:PRINT #2,j:LOCATE
#2,4,3+j:PRINT #2,"*   ="
10780 LOCATE #3,2,3+j:PRINT #3,j:LOCATE
#3,4,3+j:PRINT #3,"*   ="
10790 NEXT j
10800 LOCATE #2,2,13:PRINT #2,"10*   ="
10810 LOCATE #3,2,13:PRINT #3,"10*   ="
10900 RETURN
11000 REM > UP Würfeln <
11010 RANDOMIZE TIME
11020 w1=INT(RND(1)*6)+1
11030 w2=INT(RND(1)*6)+1
11040 w3=INT(RND(1)*6)+1
11050 w=w1+w2+w3
11100 t$=INKEY$
11110 w7=INT(RND(1)*6)+1
11120 w8=INT(RND(1)*6)+1
11130 w9=INT(RND(1)*6)+1
11190 FOR x=4 TO 6:FOR y=3 TO 5:LOCATE #
1,x,y:PRINT #1," ":NEXT y:NEXT x
11200 IF w7=1 THEN dx=0:GOSUB 12100
11210 IF w7=2 THEN dx=0:GOSUB 12200
11220 IF w7=3 THEN dx=0:GOSUB 12300
11230 IF w7=4 THEN dx=0:GOSUB 12400
11240 IF w7=5 THEN dx=0:GOSUB 12500
11250 IF w7=6 THEN dx=0:GOSUB 12600
11290 FOR x=11 TO 13:FOR y=3 TO 5:LOCATE
 #1,x,y:PRINT #1," ":NEXT y:NEXT x
11300 IF w8=1 THEN dx=7:GOSUB 12100
11310 IF w8=2 THEN dx=7:GOSUB 12200
11320 IF w8=3 THEN dx=7:GOSUB 12300
11330 IF w8=4 THEN dx=7:GOSUB 12400
11340 IF w8=5 THEN dx=7:GOSUB 12500
11350 IF w8=6 THEN dx=7:GOSUB 12600
11390 FOR x=18 TO 20:FOR y=3 TO 5:LOCATE
 #1,x,y:PRINT #1," ":NEXT y:NEXT x
11400 IF w9=1 THEN dx=14:GOSUB 12100
11410 IF w9=2 THEN dx=14:GOSUB 12200
11420 IF w9=3 THEN dx=14:GOSUB 12300
11430 IF w9=4 THEN dx=14:GOSUB 12400
11440 IF w9=5 THEN dx=14:GOSUB 12500
```

Programm
GOLD10.GEM

```
11450 IF w9=6 THEN dx=14:GOSUB 12600
11500 IF t$="" THEN 11100
11590 FOR x=4 TO 6:FOR y=3 TO 5:LOCATE #
1,x,y:PRINT #1," ":NEXT y:NEXT x
11600 IF w1=1 THEN dx=0:GOSUB 12100
11610 IF w1=2 THEN dx=0:GOSUB 12200
11620 IF w1=3 THEN dx=0:GOSUB 12300
11630 IF w1=4 THEN dx=0:GOSUB 12400
11640 IF w1=5 THEN dx=0:GOSUB 12500
11650 IF w1=6 THEN dx=0:GOSUB 12600
11690 FOR x=11 TO 13:FOR y=3 TO 5:LOCATE
 #1,x,y:PRINT #1," ":NEXT y:NEXT x
11700 IF w2=1 THEN dx=7:GOSUB 12100
11710 IF w2=2 THEN dx=7:GOSUB 12200
11720 IF w2=3 THEN dx=7:GOSUB 12300
11730 IF w2=4 THEN dx=7:GOSUB 12400
11740 IF w2=5 THEN dx=7:GOSUB 12500
11750 IF w2=6 THEN dx=7:GOSUB 12600
11790 FOR x=18 TO 20:FOR y=3 TO 5:LOCATE
 #1,x,y:PRINT #1," ":NEXT y:NEXT x
11800 IF w3=1 THEN dx=14:GOSUB 12100
11810 IF w3=2 THEN dx=14:GOSUB 12200
11820 IF w3=3 THEN dx=14:GOSUB 12300
11830 IF w3=4 THEN dx=14:GOSUB 12400
11840 IF w3=5 THEN dx=14:GOSUB 12500
11850 IF w3=6 THEN dx=14:GOSUB 12600
11900 RETURN
12000 REM Würfelaugen ausgeben
12100 LOCATE #1,5+dx,4:PRINT #1,CHR$(202
):RETURN
12200 LOCATE #1,4+dx,5:PRINT #1,CHR$(202
)
12210 LOCATE #1,6+dx,3:PRINT #1,CHR$(202
):RETURN
12300 LOCATE #1,4+dx,5:PRINT #1,CHR$(202
)
12310 LOCATE #1,5+dx,4:PRINT #1,CHR$(202
)
12320 LOCATE #1,6+dx,3:PRINT #1,CHR$(202
):RETURN
12400 LOCATE #1,4+dx,3:PRINT #1,CHR$(202
)
12410 LOCATE #1,4+dx,5:PRINT #1,CHR$(202
)
12420 LOCATE #1,6+dx,3:PRINT #1,CHR$(202
)
12430 LOCATE #1,6+dx,5:PRINT #1,CHR$(202
):RETURN
12500 LOCATE #1,4+dx,3:PRINT #1,CHR$(202
)
12510 LOCATE #1,4+dx,5:PRINT #1,CHR$(202
)
12520 LOCATE #1,5+dx,4:PRINT #1,CHR$(202
)
12530 LOCATE #1,6+dx,3:PRINT #1,CHR$(202
)
12540 LOCATE #1,6+dx,5:PRINT #1,CHR$(202
):RETURN
12600 LOCATE #1,4+dx,3:PRINT #1,CHR$(202
)
```

Programm
GOLD10. GEM

```
12610 LOCATE #1,4+dx,5:PRINT #1,CHR$(202
)
12620 LOCATE #1,4+dx,4:PRINT #1,CHR$(202
)
12630 LOCATE #1,6+dx,4:PRINT #1,CHR$(202
)
12640 LOCATE #1,6+dx,3:PRINT #1,CHR$(202
)
12650 LOCATE #1,6+dx,5:PRINT #1,CHR$(202
):RETURN
13000 REM > UP Entscheidungsalgorithmus
<
13010 k=0
13020 FOR j=0 TO 9
13030 IF x(j)<>0 GOTO 13060
13040 IF k<>0 GOTO 13060
13050 k=1:p=j:GOSUB 13100
13060 NEXT j
13070 IF k=0 THEN FOR i=0 TO 9:x(i)=x(i)
-1:NEXT i:GOTO 13020
13100 IF c(j)=0 THEN c(j)=w:RETURN
13110 k=0:RETURN
60000 DATA 176,288,400
```

Programmbeschreibung
GOLD10. GEM

Das Programm ist so lang ausgefallen, weil besonderer Wert auf eine schöne Bildschirmgrafik gelegt wurde. **Zeile 10** springt in das Unterprogramm ab **Zeile 10000**, wo die Gestaltung programmiert ist. Dort werden verschiedene Bildschirmfenster definiert und Farbwerte bestimmt. In hochauflösender Grafik werden die Umrißlinien der drei Würfel gezogen und Flächen farbig gefüllt. Ab **Zeile 10400** werden ASCII-Zeichen zu deutschen Sonderzeichen umdefiniert, und ASCII 202 wird zu einem Würfelauge gestaltet. Jeder der drei Würfel bekommt anfangs sechs Augen. In Fenster #2 wird die Wertungstabelle für den Anwender, in Fenster #3 die für den Computer geschrieben.
Die **Zeilen 30 und 40** enthalten die Aufforderung an den Spieler, eine Taste zu drücken. In **Zeile 50** wird das Unterprogramm ab **Zeilennummer 11000** angesprungen, wo der Rechner würfelt.
In den **Zeilen 11020 bis 11040** wird das Würfeln durch die Funktion RND simuliert, die drei Werte von 1 bis 6 ermittelt. In **Zeile 11050** wird sogleich ihre Summe berechnet. **Zeile 11100** übernimmt die Tastatureingabe des Benutzers, die das Würfeln beendet.
Um auf dem Bildschirm einen optischen Würfeleffekt zu erzeugen, bei dem der Benutzer aber nicht erkennen soll, was tatsächlich gewürfelt würde, werden in den **Zeilen 11110 bis 11130** drei andere Zufallszahlen ermittelt und anhand dieser Werte entsprechende Würfelaugen auf dem Monitor gezeigt.

Programmbeschreibung GOLD10. GEM

Die **Zeilen 11190 bis 11450** setzen die Zufallszahl um. Die Variable dx wirkt auf die Bildschirmposition in Fenster # 1. Das jeweilige Sprungziel von **Zeile 12100 bis 12600** führt in einen Programmteil, in dem die entsprechenden Würfelaugen mit Hilfe von dx an die richtige Bildschirmposition geschrieben werden.
Zeile 11500: Solange der Benutzer keine Taste gedrückt hat, erfolgt ein Rücksprung nach **Zeile 11100**, wo neue Zufallszahlen ermittelt und neue Würfelaugen auf dem Bildschirm ausgegeben werden. Über die drei abgebildeten Würfel flackern also ständig wechselnde Augenmuster, bis endlich eine Taste gedrückt wird. Die Bearbeitung des Programms wird danach bei **Zeile 11590** fortgesetzt.
Der Benutzer hat durch Betätigen einer Taste gewürfelt. Die **Zeilen 11590 bis 11850** bewirken nun, daß die tatsächlichen Würfel-Zufallszahlen in Augenmuster umgesetzt und auf den Bildschirm gebracht werden. Jetzt sieht der Spieler, was im Moment seines Tastendrucks wirklich gewürfelt wurde. Es erfolgt ein Rücksprung in **Zeile 11900**.
Der Benutzer ist nun gefragt, an welcher Position seiner Wertungstabelle er den Wurf einordnen will (**Zeilen 100 bis 120**). Es wird geprüft, ob die Eingabe zulässig ist (**Zeilen 130 bis 140**), oder ob der Benutzer die eingegebene Plazierung in seiner Wertungstabelle in einem früheren Spielzug schon besetzt hat (**Zeile 150**).
Ist die Eingabe zulässig, wird in **Zeile 160** der Feldvariablen u der gewürfelte Wert zugeordnet und in der Variablen uu der damit erreichte Zwischenpunktestand festgehalten. Die **Zeilen 170 und 180** schreiben das Ergebnis auf dem Bildschirm in die Wertungstabelle des Anwenders.
Jetzt ist der Computer an der Reihe. In **Zeile 200** legt er sich in der Feldvariablen x eine Tabelle zurecht, welche die gewürfelten Augen w in Relation zu den möglichen Würfelwerten von 3 bis 18 setzt. Je größer ein Wert x(j) ist, desto weiter ist er von der Idealposition entfernt. Im Unterprogramm ab **Zeile 13000** wird ermittelt, an welche noch freie Stelle der Wertungstabelle, die dieser Idealposition möglichst nahe kommt, der Würfelwert w gesetzt werden kann.
Für diese Entscheidung benötigt das Programm ganze neun BASIC-Zeilen. Dabei wird zuerst nach einem Wert x(j) = 0 gesucht, denn der markiert die optimale Position. Findet die FOR-NEXT-Schleife von **Zeile 13020 bis 13060** einen solchen Wert, wird in **Zeile 13050** die Variable k auf 1 gesetzt, damit die Schleife weiter bearbeitet werden kann, ohne daß das gefundene Ergebnis wieder

Programmbeschreibung
GOLD10. GEM

verloren geht. Bei späteren Durchläufen können nämlich zwei x(j) den Wert 0 haben. Außerdem wird der augenblickliche Wert des Schleifenzählers j in der Variablen p verwahrt, womit die gefundene Position in der Wertungstabelle numerisch erfaßt ist.
Nachdem eine mögliche Position gefunden ist, muß überprüft werden, ob sie nicht schon bei einem früheren Spielzug belegt wurde. Das geschieht im Unterprogramm ab **Zeile 13100**.
In der Feldvariablen c werden die vom Computer vollzogenen Züge erfaßt. Wenn die durch die gefundene Position p markierte Variable c(p) noch den Wert 0 hat, also noch nicht mit einem Wurf belegt wurde, wird ihr der aktuelle Wurf w zugeordnet. Ist das aber nicht der Fall, wird die Flag-Variable k wieder auf 0 gesetzt.
Der Rücksprung führt nach **Zeile 13070**. Wenn k = 0 wahr ist, konnte der Wurf w der Variablen c(p) nicht zugeordnet werden, weil sie bereits einen Wert enthält. Deshalb werden jetzt alle Werte der Feldvariablen x um den Wert 1 verringert. Durch dieses Herabsetzen der x(i)-Werte erhalten die beiden, der optimalen am nächsten liegenden Positionen irgendwann den Wert 0. Sie werden von der FOR-NEXT-Schleife gefunden, und im Unterprogramm wird überprüft, ob sie noch frei sind. Dieser Vorgang wird so lange wiederholt, bis eine freie und damit der idealen weitestgehend angenäherte Position gefunden wurde.
Eigentlich müßte das bei **Zeile 13000** beginnende Unterprogramm in einer **Zeile 13080** mit RETURN beendet werden. Wenn eine freie Position gefunden wurde, hat k den Wert 1, und die **Zeile 13070** wird nicht bearbeitet, weil die IF-Bedingung nicht erfüllt ist. Der Rechner macht mit **Zeile 13100** weiter, in der eigentlich das Unterprogramm zur Überprüfung von c(p) beginnt. Da c(p) inzwischen der Wert w zugewiesen wurde, wird aber auch diese Zeile nicht bearbeitet. In **Zeile 13110** wird daraufhin (überflüssigerweise) k auf 0 gesetzt, und dann findet BASIC das notwendige RETURN, das ins Hauptprogramm zurückführt.
Eine Anmerkung noch: ist die optimale Position nicht mehr frei, gibt es im folgenden meist zwei Positionen, die von der optimalen gleich weit entfernt sind. Das Programm findet immer die mit dem geringeren Stellenwert, die also in der Wertungstabelle weiter oben steht. Man könnte auch zufällig eine der beiden gleichwertigen Positionen auswählen. Statistisch betrachtet würde dabei aber kein besseres Ergebnis erzielt werden, da beide Positionen im Moment der Entscheidungsnotwendigkeit gleichwertig sind.

Programmbeschreibung GOLD10. GEM

Die Bearbeitung des Hauptprogramms wird in **Zeile 210** fortgesetzt. Die Zwischensumme des Computerergebnisses wird in cc festgehalten und in den **Zeilen 220 und 230** in das Fenster #3 in die Wertungstabelle des Computers geschrieben.
Zeile 300 wacht darüber, daß der ganze Vorgang zehnmal wiederholt wird. Dann ist das Spiel beendet; in den Zeilen **310 bis 360** wird der Sieger ermittelt und verkündet. Die **Zeilen 370 bis 400** ermöglichen es dem Anwender, durch Betätigen einer beliebigen Taste ein neues Spiel zu beginnen.
Offen gesagt, dieses Spiel ist nicht besonders intelligent. Deshalb sollten Sie aber nicht gleich das Buch in die Ecke werfen. Zwei Kapitel weiter werden Sie einem Spiel begegnen, von dem der Rechner nur die Regeln kennt. Das erste Spiel beginnt er also völlig »dumm«, er lernt aber Partie um Partie dazu, bis Sie wirklich keine Chance mehr haben, gegen ihn zu gewinnen.
Auch bei GOLD10.GEM sind die Gewinnaussichten für den menschlichen Spieler gering, jedoch nicht Null. Ein bißchen Zufall kommt dadurch ins Spiel, daß Spieler und Computer im Rückblick auf alle zehn gewürfelten Zahlen mehr oder weniger glückliche Zuordnungen getroffen haben. Außerdem berücksichtigt der Entscheidungsalgorithmus des Programms nicht die statistische Mengenverteilung der verschiedenen möglichen Würfelergebnisse.

Programmverbesserung

Das Programm geht davon aus, daß die Augensumme von drei Würfeln im Mittel 10,5 Punkte beträgt. Der niedrigstmögliche Wurf ist 3, der höchste 18. Würden die Augen der zehn Würfe des Spiels gleichmäßig verteilt fallen, ergäbe sich eine theoretische Wertereihe von (gerundet): 3; 4,6; 6,3; 8; 9,6; 11,3; 13; 14,6; 16,3 und 18. Im Verlauf der Partie müßte nur noch für den jeweiligen Wurf aus dieser optimalen Reihe der Wert mit der geringsten Abweichung gefunden und der Wurf in die entsprechende Zeile eingetragen werden. Die **Programmzeile 200** geht von dieser Idealverteilung aus.
Die Häufigkeitsverteilung der Ergebnisse von drei Würfeln ist aber nicht gleichmäßig. Unter den 6*6*6 (= 216) möglichen Würfen sind die Werte 3 und 18 nämlich statistisch nur je einmal vertreten, während die Werte 10 oder 11 gleich 27mal fallen.
Wenn Sie an der exakten Häufigkeitsverteilung interessiert sind, wollen Sie das Programm GOLD10.GEM vielleicht verbessern.

```
1 REM DICEP.DEM
10 DIM x(18)
100 FOR a=1 TO 6
110 FOR b=1 TO 6
120 FOR c=1 TO 6
200 x=a+b+c
210 FOR j=3 TO 18
220 IF j=x THEN x(j)=x(j)+1
230 NEXT j
300 NEXT c
310 NEXT b
320 NEXT a
400 CLS
410 FOR j=3 TO 18
420 PRINT USING"##";j;:PRINT":";:PRINT U
SING"###";x(j)
430 NEXT j
```

Programm DICEP.DEM

Das Auge des Computers

Wahrnehmen, unterscheiden, erkennen ist die Bedeutung des lateinischen Wortes *intellegere*. Die im Gehirn gewonnene Erkenntnis setzt Daten voraus, die über die Sinnesorgane aufgenommen werden. Das ist bei einem Computer nicht anders als bei Menschen. Nur besteht der Kontakt des Rechners zur Außenwelt fast durchgehend aus Benutzereingaben, die über die Tastatur erfolgen, oder im Einlesen solcher Eingaben von einem Massenspeicher.

Theoretisch können alle möglichen Außenweltdaten in schriftlicher Form an den Rechner übergeben werden. Doch ist schon die exakte Beschreibung einer geometrischen Form, zum Beispiel eines nach technischen Gesichtspunkten gestalteten Gebrauchsgegenstandes, so umfangreich, daß es erstrebenswert ist, die Maschine mit der ihr eigenen ermüdungsfreien Ausdauer in die Lage zu versetzen, diese Daten selbst zu ermitteln.

Vom Modell zur dreidimensionalen Konstruktionszeichnung

Bei der Konstruktion von Kraftfahrzeugen werden robotergesteuerte Tastarme verwendet, die ein von Designern gebautes Modell abtasten und die gefundenen Raumkoordinaten dem Rechner in Form von Daten übergeben. Der Computer entwickelt daraus ein internes Modell, das als dreidimensionale Konstruktionszeichnung auf dem Monitor nicht nur drehend, wendend und im Ausschnitt betrachtet, sondern im weiteren Verlauf der Entwicklungsarbeit auch verändert werden kann.

Während abtastende Verfahren weitgehend ausgereift sind und überall in Industrie und Forschung eingesetzt werden, steckt die Bildverarbeitung noch in den Anfängen. Relativ gering waren die Probleme, Bildvorlagen in numerische Daten umzusetzen, sie zu digitalisieren. Hier gibt es eine lange Tradition, die beim Holzschnitt im klassischen Tibet beginnt, über Rasterung und Farbtrennung verläuft und in der analogen Bildverarbeitung der Fernsehtechnik endet. Die vorhandenen Videodaten mußten lediglich in eine digitale Form gebracht werden.

Digitalisieren und Scannen

Für die Aufnahme der Bildvorlage bieten sich zwei Wege an. Eine natürliche Szene kann mit einer Videokamera erfaßt und von einem Videodigitizer in Zahlenwerte umgewandelt werden. Oder eine Printvorlage (Foto, Gemälde etc.) wird von einem Scanner Zeile für Zeile und Punkt für Punkt abgetastet. Dabei werden die Positionen und Farbwerte ermittelt und aufgezeichnet.

Sind die Daten eines Bildes erst einmal erfaßt, kann man mit diesem Material unbegrenzt spielen, wie die moderne Fernsehdramaturgie täglich beweist. Durch entsprechende Programme verarbeitet, können die Farben des Bildes verändert werden. Es kann gedehnt, gestaucht, verzerrt oder zerstückelt werden. Die Bilddaten können auf die Oberfläche anderer Körper (Kugeln, Quader etc.) projiziert und diese Formen beliebig über die Bildfläche bewegt werden. So entstehen dreidimensionale Effekte, zum Beispiel des Wegfliegens oder Explodierens.

Erkennen kann der Computer dabei aber keinen einzigen Gegenstand. Er verarbeitet lediglich die Daten räumlicher Koordinaten und wendet auf sie geometrische Funktionen an. In einem digital erfaßten Bild auch nur den Gegenstand vom Hintergrund abzugrenzen, stellt heute noch die Grenzmarke der Entwicklung dar. Gewisse Erfolge sind inzwischen zu verzeichnen, wenn der zu erkennende Gegenstand dem Programm in Form von Daten bekannt ist. Robotersysteme sind bereits in der Lage, Werkstücke, die beliebig auf ein Fließband fallen, zu lokalisieren und anschließend so gleichmäßig auszurichten, daß sie maschinell weiterverarbeitet werden können.

Wahrnehmung besteht für ein Computersystem also immer darin, über ein Peripheriegerät optische, akustische oder sonstige Informationen aufzunehmen und in Daten umzuwandeln, die es mit Hilfe der entsprechenden Programme weiterverarbeiten kann.

Das ist beim menschlichen Gehirn im Prinzip nicht anders. Unsere Sinnesorgane wandeln ebenfalls Umweltreize in Daten um, die von den Nerven an das Gehirn zur Verarbeitung übergeben werden. Die Biotechnologie unseres Gehirns ist allerdings wesentlich weiter entwickelt. Gewaltige Datenmengen können in kürzesten Zeiträumen übergeben und verarbeitet werden.

Das Aktionsprinzip des Computers anhand eines einfachen Programmes: FANGEN.DEM

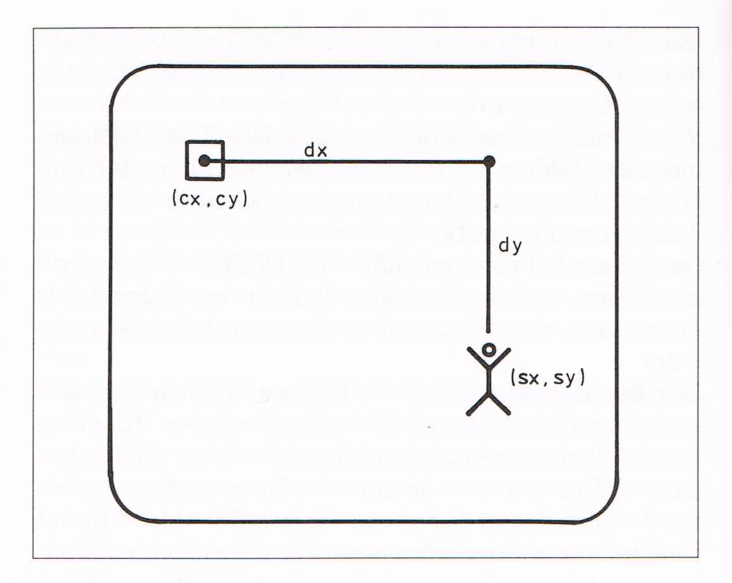
Das folgende Programm FANGEN.DEM zeigt in simpelster Weise, wie der Computer in Form von Daten wahrnimmt und daraus eine Handlungsentscheidung ableitet.

Der Benutzer findet sich als kleines Männchen in der unteren rechte Ecke des Bildschirms wieder. Über den Joystick kann er seinen Spieler auf der Mattscheibe bewegen. Der Rechner steuert den kleinen Kasten, der bei Spielbeginn in der oberen linken Ecke steht. Er hat den Auftrag, sich auf kürzestem Weg dem Spieler zu nähern und ihn zu fangen, indem er den Kasten auf die gleiche Bildschirmposition bringt, auf der sich der Spieler gerade befindet. Die Position des Spielers wird in den Variablen sx und sy verfolgt und mit ihrer Hilfe auf dem Monitor dargestellt. Die Position des Fängers wird von cx und cy beschrieben.

Da das Programm nur zur Demonstration gedacht ist, sei hier eine Abkürzung erlaubt, schon weil damit aufwendige Technik umgangen wird, die nicht selbstverständlich vorhanden ist. Natürlich könnte mit entsprechendem Aufwand die Bewegung (Positionsänderung) des Spielers auf dem Bildschirm von einer Videokamera aufgenommen, das digitalisierte Bild nach dem bekannten Muster des Spielermännchens abgesucht und die entsprechende Lokalisierung in Bildschirmkoordinaten umgerechnet werden.

Wenn von diesem Programm ganz einfach die beiden Variablen sx und sy gelesen werden, dann schaut es dem Spieler nicht etwa in die Karten, sondern es simuliert einen reichlich komplizierten Vorgang in der denkbar einfachsten Form.

Hat der Computer die Position des Spielers gelesen, setzt er sie zu der seines Fängers in Beziehung und bewegt ihn so um eine Schreibstelle auf dem Bildschirm weiter, daß die Entfernung verringert wird. Zu diesem Zweck vergleicht das Programm die X-Koordinaten (sx und cx) und Y-Koordinaten (sy und cy) von Spieler und Fänger, errechnet den Abstand (dx und dy) zwischen beiden und verändert dann die Fängerposition (cx, cy) entsprechend.



Vorsicht! FANGEN.DEM ist kein Spiel. Selbst die flinkesten Joystick-Finger haben gegen die gnadenlose Präzision des Programms nur eine frustrierend geringe Chance. Nach wenigen Schritten ist das Spielmännchen gefangen und reagiert nicht mehr auf Steuerungsversuche des Spielers.

Programm FANGEN.DEM

```
 1 REM FANGEN.DEM
 10 MODE 1:GOSUB 60010
 20 cx=1:cy=1:sx=40:sy=25
 30 LOCATE cx,cy:PEN 3:PRINT CHR$(201);
 40 LOCATE sx,sy:PEN 1:PRINT CHR$(200);
 100 REM > Spielerzug <
 110 ex=0:ey=0
 120 s=JOY(0)
 130 IF s=0 GOTO 1010
 140 IF (s AND 1)=1 THEN ey=-1
 150 IF (s AND 2)=2 THEN ey=1
 160 IF (s AND 4)=4 THEN ex=-1
 170 IF (s AND 8)=8 THEN ex=1
 180 IF sx+ex<1 OR sx+ex>40 GOTO 1010
 190 IF sy+ey<1 OR sy+ey>25 GOTO 1010
 200 LOCATE sx,sy:PEN 1:PRINT " ";
 210 sx=sx+ex:sy=sy+ey
 220 LOCATE sx,sy:PRINT CHR$(200);
 1000 REM > Computerzug <
 1010 LOCATE cx,cy:PEN 3:PRINT " ";
 1020 vx=SGN(sx-cx):vy=SGN(sy-cy)
 1030 dx=ABS(sx-cx):dy=ABS(sy-cy)
 1040 IF dx>dy THEN cx=cx+vx*1
 1050 IF dx<dy THEN cy=cy+vy*1
 1060 IF dx=dy THEN cx=cx+vx*1:cy=cy+vy*1
 1070 LOCATE cx,cy:PRINT CHR$(201);
 1080 IF sx=cx AND sy=cy GOTO 2000
```

Programm
FANGEN.DEM

```
1090 GOTO 110
2000 LOCATE sx,sy:PEN 1:PRINT CHR$(202);
2010 FOR w=0 TO 200:NEXT w
2020 LOCATE sx,sy:PEN 3:PRINT CHR$(203);
2030 FOR w=0 TO 150:NEXT w
2040 GOTO 2000
60000 REM > Sonderzeichen <
60010 SYMBOL AFTER 200
60020 SYMBOL 200,96,96,16,232,24,39,32,3
2
60030 SYMBOL 201,255,129,129,129,129,129
,129,255
60040 SYMBOL 202,153,90,36,24,24,36,66,1
29
60050 SYMBOL 203,255,165,255,165,165,255
,165,255
60060 RETURN
```

Programmbeschreibung
FANGEN.DEM

In **Programmzeile 10** wird der Bildschirmmodus 1 für Textdarstellung mit 40 Zeichen pro Zeile gewählt. Danach erfolgt eine Verzweigung in das Unterprogramm ab **Zeile 60010**, wo die ASCII-Zeichen 200 bis 203 in ein stehendes und ein laufendes Männchen, einen Kasten und ein Gitter umdefiniert werden.
Zeile 30 stellt den Kasten mit Farbe 3 für den Computer in die linke obere Ecke, **Zeile 40** das laufende Männchen in Farbe 1 für den Spieler in die untere rechte Ecke des Bildschirms.
Der Spieler hat den ersten Zug. **Zeile 120** nimmt den Zustand des Joysticks auf und übergibt ihn an die Variable s. Bewegt sich der Spieler nicht, verzweigt **Zeile 130** nach **Zeile 1010**, wo der Computer seinen Kasten bewegt. Sonst setzen die **Zeilen 140 bis 170** die Joystickeingabe in Bewegungswerte um.
Versucht der Spieler, über den linken oder rechten Rand (**Zeile 180**) oder über den oberen oder unteren Rand (**Zeile 190**) hinauszuziehen, wird an den Computer übergeben.
Bei einer zulässigen Bewegung des Spielers wird die alte Position in **Zeile 200** gelöscht. **Zeile 210** verändert die Position (sx, sy) der eingegebenen Bewegung entsprechend. Und **Zeile 200** schreibt das Männchen an die neue Position.
Der Rechner beginnt seinen Zug damit, daß er seine alte Position (cx ,cy) in **Zeile 1010** löscht. In **Zeile 1020** wird festgestellt, ob sich das Spielmännchen rechts oder links vom Fänger befindet. Steht es rechts, ist der Wert sx kleiner als cx, das Vorzeichen ein Minus. Die Funktion SGN gibt der Variablen vx dann den Wert —1. Auf die gleiche Weise wird in vy festgehalten, ob sich das

Programmbeschreibung
FANGEN.DEM

Männchen ober- oder unterhalb des Fängers befindet. Für die bestehende Version des Programms ist diese Feststellung irrelevant, weil der Spieler gar keine Chance hat, sein Männchen am Fänger vorbei und in eine Position links oder oberhalb von ihm zu bringen. Die Fluchtmöglichkeiten des Spielers lassen sich aber verbessern, wenn man ihm die Chance gibt, das Spielmännchen über die Bildschirmränder hinweglaufen und an der jeweils gegenüberliegenden Seite wieder im Spielfeld erscheinen zu lassen. Dazu müßten die **Zeilen 180 und 190** geändert werden. Versorgt man dann aber nicht auch den Fänger mit dieser Zugmöglichkeit, kann sich der Spieler an den Rand der Fläche stellen und das Programm durch fortgesetztes Hin- und Herspringen beliebig lange narren.

In **Zeile 1030** wird der absolute Betrag der Entfernung vom Fänger zum Spielmännchen errechnet und in dx und dy notiert. Ist die Entfernung in X-Richtung größer, bewegt sich der Fänger (**Zeile 1040**) einen Schritt in diese Richtung. Ist die Entfernung in Y-Richtung größer, bewegt er sich (**Zeile 1050**) einen entsprechenden Schritt auf das Männchen zu. Ist die Entfernung in beiden Dimensionen gleich groß (**Zeile 1060**), verringert das Programm die Entfernung in beiden Richtungen um eine Schreibstelle, der Fänger bewegt sich also diagonal. Abschließend wird in **Zeile 1070** die neue Position des Fängers auf den Bildschirm geschrieben.

Sind nach dem Computerzug die Positionen des Spielers und des Fängers identisch, verzweigt das Programm in **Zeile 1080** nach **Zeile 2000**, sonst führt **Zeile 1090** nach **Zeile 110** zurück und übergibt die Kontrolle damit wieder an den Joystick des Anwenders.

Programmzeilen 2000 bis 2040: Ist das Spielmännchen gefangen, wird in dieser endlosen Schleife an die entsprechende Position abwechselnd ein stehendes Männchen und ein Gitter geschrieben. Das Programm muß durch Betätigen der Taste <ESC> abgebrochen werden.

Das in der vorliegenden Version stark reduzierte Wahrnehmen des Rechners könnte ein wenig realistischer gestaltet werden, wenn sich außer dem Spielmännchen weitere Figuren, eventuell zufällig, über den Bildschirm bewegen würden, die sich in Form und/oder Farbe von ihm unterscheiden. Der Computer könnte dann den Bildschirm Schreibstelle um Schreibstelle abtasten und vorgefundene Figuren daraufhin überprüfen, ob es sich bei ihnen um das Spielmännchen handelt. Dazu müßte das Programm mit Daten versorgt werden, die es ihm ermöglichen, das Männchen zu identifizeren. In der ein-

fachsten Form verfügt es über die vollständigen Daten, es kennt also das Muster des zu erkennenden Objekts. Bei fortgeschritteneren Verfahren genügen dem Programm einzelne Merkmale.

Mustererkennung: Zunächst ganz einfach

Der Forschungsbereich, der sich mit dieser Aufgabenstellung beschäftigt, wird deshalb Mustererkennung genannt. Schwerpunkte der Bemühungen liegen heute im Erkennen von Objekten und von menschlicher Sprache. Bei der simpelsten Methode der Mustererkennung wird mit einer Maske gearbeitet, die an den zu erkennenden Gegenstand gelegt wird. Das Programm überprüft, ob sich Gegenstand und Maske decken.
Das folgende Programm MUSTER.DEM erfragt als Gegenstand einen String t$, in dem nach einem anderen String x$ gesucht werden soll. Dieser andere String wird als Maske Zeichen um Zeichen über den Gegenstand t$ gelegt.

Programm MUSTER.DEM

```
1 REM MUSTER.DEM
10 CLS:CLEAR:PRINT"BITTE ZU DURCHSUCHEND
EN TEXT EINGEBEN":PRINT:PRINT
20 INPUT t$
30 PRINT:PRINT:PRINT"BITTE ZU SUCHENDEN
STRING EINGEBEN":PRINT:PRINT
40 INPUT x$
100 IF t$=x$ THEN PRINT"DER GESUCHTE STR
ING IST: ":PRINT:PRINT:PRINT T$:GOTO 100
0
110 x=LEN(x$):t=LEN(t$):d=t-x:DIM p(d+1)
120 IF d<0 THEN PRINT"ZU SUCHENDER STRIN
G IST ZU LANG":PRINT:PRINT:GOTO 1000
130 FOR j=1 TO d+1
140 s$=MID$(t$,j,x)
150 IF s$=x$ THEN p(j)=j
160 NEXT j
200 PRINT:PRINT:PRINT"DER GESUCHTE STRIN
G BEFINDET SICH AN":PRINT
210 FOR j=1 TO d+1
220 IF p(j)>0 THEN PRINT"POSITION ";:PRI
NT USING"###";j;:PRINT" BIS ";:PRINT USI
NG"###";(j+x-1):f=1
230 NEXT J
240 IF f=0 THEN PRINT"KEINER POSITION"
250 PRINT:PRINT"IM VORGEGEBENEN TEXT"
1000 PRINT:PRINT:PRINT:PRINT"WIEDERHOLEN
: BITTE TASTE DRUECKEN"
1010 IF INKEY$="" GOTO 1010
1020 GOTO 10
```

Programmbeschreibung MUSTER.DEM

In **Zeile 10** werden der Bildschirm und alle Variablen gelöscht, um das System in einen unberührten Zustand zurückzusetzen. Dann wird in **Zeile 20** nach dem zu

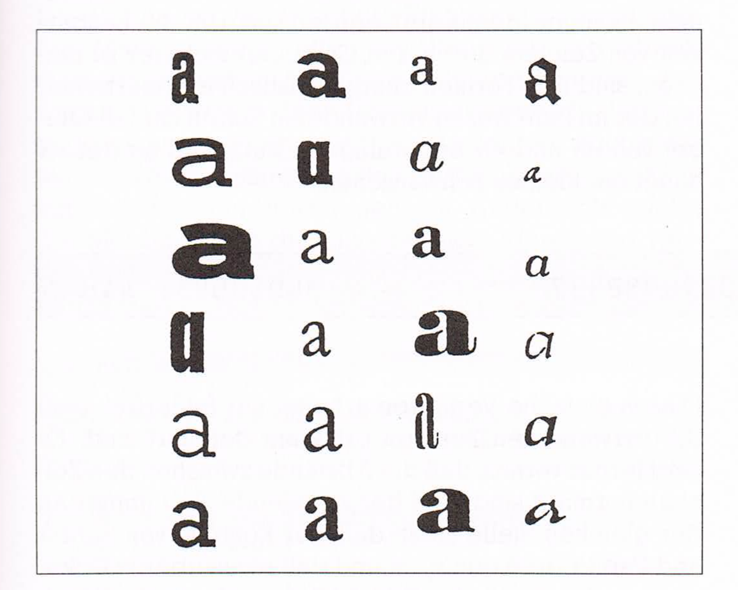
Programmbeschreibung
MUSTER.DEM

durchsuchenden Gegenstand t$ gefragt. Tippen Sie einfach blind auf die Tastatur. Bis zu 255 Zeichen kann t$ aufnehmen.
Zeilen 30 und 40 fragen dann, nach welchem String gesucht werden soll. Auch hier können Sie eine beliebige Folge von Zeichen eingeben. Wenn aber wirklich etwas gefunden werden soll, wird es gut sein, x$ nur mit zwei, drei Zeichen zu füllen.
Das Programm überprüft zuerst, ob Gegenstand und Maske womöglich identisch sind (**Zeile 100**) und gibt gegebenenfalls eine entsprechende Meldung aus.
Als nächstes wird die Länge der beiden eingegebenen Strings ermittelt, ihre Differenz berechnet und eine Feldvariable p entsprechend dimensioniert, in der später die Fundstellen vermerkt werden.
Wenn die Maske größer als der Gegenstand ist, kann natürlich nichts gefunden werden. **Zeile 120** schreibt eine entsprechende Meldung auf den Bildschirm.
Bei **Zeile 130** beginnt die Suche. Die FOR-NEXT-Schleife bestimmt als Anfangswert für den Schleifenzähler 1, weil die Zeichen in einem String mit 1 beginnend gezählt werden. Das Ende des Schleifenzählers wird durch das erste Zeichen, das hinter der Längendifferenz des Gegenstandes und der Maske liegt, markiert. Dies ist das letzte Zeichen, bei dem, beginnend im Gegenstand, noch genügend Platz für die Maske verbleibt.
Zeile 140 greift aus dem Gegenstand t$ eine Zeichenfolge heraus, die der Länge der Maske entspricht. Dies geschieht bei der Position j, die bei jedem Durchlauf um 1 erhöht wird. Der Teilstring aus t$ wird s$ zugeordnet.
Wenn der so gefundene Teilstring s$ identisch mit der Maske x$ ist, wird in **Zeile 150** die Position im Gegenstand (j) in der Feldvariablen p notiert. Dieses einzelne Erfassen der Fundstelle ist deshalb notwendig, weil auch mehrere Teile des Objekts mit der Maske übereinstimmen können.
Von **Zeile 200** an wird das Ergebnis der Suche auf den Monitor geschrieben. Die Schleife ab **Zeile 210** geht alle möglichen Positionen im Gegenstand (1 bis d + 1) durch. Wenn die entsprechende Variable p(j) einen Wert ungleich 0 enthält, erfolgt in **Zeile 220** eine Bildschirmausgabe, die Start- und Endposition der Fundstelle enthält. In der Variablen f wird mit dem Wert 1 notiert, daß etwas gefunden wurde.
Wurde nämlich keine Stelle in t$ gefunden, die mit x$ identisch ist (f = 0), erfolgt in **Zeile 240** eine entsprechende Meldung.
Die **Zeilen 1000 bis 1020** fangen das Programm auf. Durch Betätigen einer beliebigen Taste wird nach **Zeile**

10 verzweigt, wo die Suche von neuem beginnen kann. Das Programm ist in der Lage, Zeichen zu erkennen. Dies kann es jedoch nur, weil die Formen der gesuchten Zeichen klar definiert sind. Stellt man sich die Eingaben nicht als Zeichenkette, sondern als gesprochene Wörter vor, wird der Vorgang schon hochkompliziert. Selbst innerhalb eines nationalen Sprachraums unterscheidet sich die Aussprache der einzelnen Menschen voneinander.

Mustererkennung: Schon bei Buchstaben wird es kompliziert

Etwas näher am vorgestellten Beispiel liegt das Erkennen von Hand geschriebener Buchstaben. Auch die Handschrift jedes Menschen sieht im Detail anders aus, obwohl eine prinzipielle Form, man spricht von Gestalt, für jedes Schriftzeichen vorgegeben ist.



Sogar gedruckte lateinische Schriften gibt es in tausenden verschiedener Schnitte. Dabei kennt das »a« gleich zwei verschiedene Gestalten: mit und ohne Bügel. Um eine fehlerfreie Computerlesbarkeit zu gewährleisten, muß heute noch mit genormten Zeichen gearbeitet werden. In der Wirtschaft ist zu diesem Zweck die ORC-Schrift eingeführt worden, die meist in Verbindung mit Bar-Codes zur Warenauszeichnung eingesetzt wird. Auf dem rückseitigen Deckel dieses Buches sehen Sie ein Beispiel dafür.

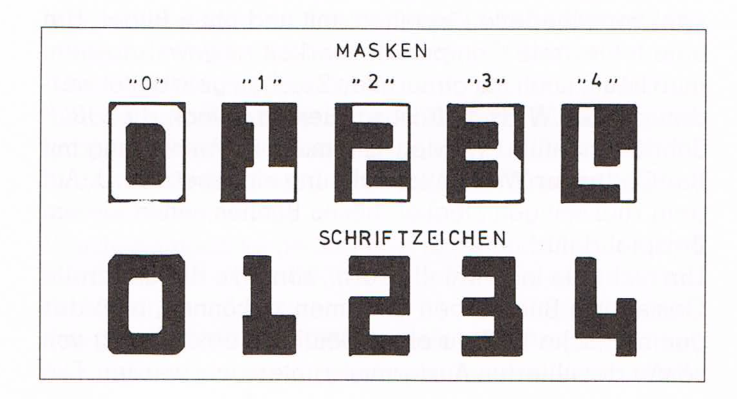
Um nicht die individuelle Form, sondern die generelle Gestalt von Buchstaben erkennen zu können, muß der geometrische Aufbau eines Zeichens unabhängig von seiner detaillierten Ausformung untersucht werden. Die

Gestalt eines jeden Buchstabens wird in Elemente wie rechte und spitze Winkel, vertikale und horizontale Linien, Schräglinien, geschlossene Bögen und Bogenteile zerlegt und die Häufigkeit eines jeden Elements ermittelt. Ein »F« enthält drei rechte Winkel, eine vertikale und zwei horizontale Linien. Besteht der zu erkennende Buchstabe aus nur zwei Bogenteilen (»S«), kann es sich nicht um ein »F« handeln.
Mit den aufgezählten Elementen sind aber nur begrenzte Unterscheidungen möglich. Ein großes »S« kann nicht von einem kleinen »s« getrennt werden. Je weiter aber die Gestalten in elementare Teile zerlegt werden, desto genauer kann eine solche Mustererkennung arbeiten.
Bei einem einfachen Erkennungsverfahren mit Masken muß der Buchstabe exakt mit der Maske übereinstimmen, wenn er identifiziert werden soll. Um die Lesbarkeit von Zeichen durch den Computer sicherer zu machen, sind ihre Formen charakteristisch ausgearbeitet. Bei der im Bankwesen verwendeten Schrift sind die Bügel von »6« und »9« beunruhigend kurz, und an der »1« hängt ein kleines Schwänzchen.

1030731900027⑈ 0276792437⑆ 360100043⑈ 11⑆

Einfache Leseverfahren setzen genaue Definitionen voraus

Dieses einfache Verfahren arbeitet nur fehlerfrei, weil die verwendeten Zeichen prägnant definiert sind. Es setzt ferner voraus, daß die Abstände zwischen den Zeichen normiert sind, daß der zu lesende Text immer an der gleichen Stelle steht, daß der Kontrast von Schrift und Papier groß genug ist und daß ein sauberes Dokument zu lesen ist, die Schriftzeichen also nicht durch Schmutzpartikel, Knicke, Risse oder andere Zeichen überlagert werden.



Ein Beispiel für eine toleranter Erkennungsmethodik

Das Programm ACTSYM.MER arbeitet nach einer etwas offeneren Methode, kann deshalb aber auch nur bedingt erkennen, welcher Buchstabe eingegeben wurde.
Nach Aufruf des Programms erscheint auf dem schwarzen Bildschirm ein weißes Fenster, in das der Benutzer 8 x 8 Punkte setzen kann. Dazu bewegt er mit dem Joystick den Cursor im Schreibfenster und setzt oder löscht Punkte mit dem Feuerknopf. Für die Form der Buchstaben gelten folgende Bedingungen: Sie müssen in Blockbuchstaben gesetzt werden, ihre Striche dürfen nur einen Punkt dick sein, und das Zeichen soll das Schreibfenster möglichst weit ausfüllen.
Ist der Buchstabe geschrieben, wird durch Drücken der Taste <E> der Identifizierungsvorgang ausgelöst. Das Programm vergleicht die Eingabe mit vorgegebenen Masken, die so offen gehalten sind, daß Abweichungen in bestimmten Grenzen toleriert werden. Bewegt sich die gefundene Übereinstimmung in den vorgegebenen Grenzen, wird der vermutete Buchstabe und die Übereinstimmung des eingegebenen Zeichens mit der Maske in Prozent genannt. Andernfalls erfolgt die Meldung, daß die Eingabe nicht identifiziert werden kann.

Programm ACTSYM.MER

```
1 REM ACTSYM.MER
10 GOSUB 10010
20 x=2:y=2:xa=32:xc=32:xt=28
100 IF (INKEY$="E") OR (INKEY$="e") GOTO
 1010
110 s=JOY(0):u=0:v=0:p=0
120 IF s=0 GOTO 100
130 IF (s AND 1)=1 THEN u=0:v=-1
140 IF (s AND 2)=2 THEN u=0:v=1
150 IF (s AND 4)=4 THEN u=-1:v=0
160 IF (s AND 8)=8 THEN u=1:v=0
170 IF (x+u)<2 OR (x+u)>9 GOTO 100
180 IF (y+v)<2 OR (y+v)>9 GOTO 100
200 LOCATE x,1:PRINT " ":LOCATE x,10:PRI
NT " ":LOCATE 1,y:PRINT " ":LOCATE 10,y:
PRINT " "
210 x=x+u:y=y+v
220 PEN 3:LOCATE x,1:PRINT CHR$(241):LOC
ATE x,10:PRINT CHR$(240):LOCATE 1,y:PRIN
T CHR$(243):LOCATE 10,y:PRINT CHR$(242)
230 IF (s AND 16)=16 THEN p=1
240 m(x-2,y-2)=m(x-2,y-2) XOR p
250 LOCATE x,y:PEN 2
260 IF m(x-2,y-2)=0 THEN PRINT CHR$(143)
270 IF m(x-2,y-2)=1 THEN PRINT CHR$(250)
280 FOR w=0 TO 100:NEXT w
290 GOTO 100
1000 REM > Auswertung <
```

Programm
ACTSYM.MER

```
1010 CLS #1:LOCATE #1,15,2:PRINT #1,"BIT
TE WARTEN"
1020 LOCATE #1,14,4:PRINT #1,"SYSTEM REC
HNET"
1100 FOR j=0 TO 7:FOR i=0 TO 7
1110 IF m(i,j)=1 THEN pn=pn+1
1120 NEXT i:NEXT j
1130 FOR j=0 TO 7
1140 IF m(0,j)=1 THEN n(j)=n(j)+128
1150 IF m(1,j)=1 THEN n(j)=n(j)+64
1160 IF m(2,j)=1 THEN n(j)=n(j)+32
1170 IF m(3,j)=1 THEN n(j)=n(j)+16
1180 IF m(4,j)=1 THEN n(j)=n(j)+8
1190 IF m(5,j)=1 THEN n(j)=n(j)+4
1200 IF m(6,j)=1 THEN n(j)=n(j)+2
1210 n(j)=n(j)+m(7,j)
1220 NEXT j
1300 FOR j=0 TO 7
1310 qa(j)=a(j) AND n(j)
1320 qc(j)=c(j) AND n(j)
1330 qt(j)=t(j) AND n(j)
1340 NEXT j
1400 FOR j=0 TO 7:FOR i=1 TO 8
1410 IF MID$(BIN$(qa(j)),i,1)="1" THEN p
a=pa+1
1420 IF MID$(BIN$(qc(j)),i,1)="1" THEN p
c=pc+1
1430 IF MID$(BIN$(qt(j)),i,1)="1" THEN p
t=pt+1
1440 NEXT i:NEXT j
1500 DEF FNpro=100*i/pn
1510 i=pa:ja=i/xa:ra=FNpro
1520 i=pc:jc=i/xc:rc=FNpro
1530 i=pt:jt=i/xt:rt=FNpro
1540 MODE 1
1550 j=MAX(ja,jc,jt):IF j<0.33 THEN GOSU
B 2000:GOTO 1660
1560 r=MAX(ra,rc,rt)
1600 PRINT"DAS EINGEGEBENE ZEICHEN":PRIN
T
1610 IF r<75 THEN PRINT"KANN NICHT EINDE
UTIG IDENTIFIZIERT":PRINT:PRINT"WERDEN.
DIE UEBEREINSTIMMUNG MIT":PRINT:PRINT"BE
KANNTEN MUSTERN IST ZU GERING,":PRINT:PR
INT"NAEMLICH NUR ETWA";ROUND(r,2);"%":GO
TO 1660
1620 PRINT"IST MIT EINER WAHRSCHEINLICHK
EIT VON CA."
1630 IF r=ra THEN PRINT ROUND(ra,2);"%":
PRINT:PRINT"DER BUCHSTABE         A"
1640 IF r=rc THEN PRINT ROUND(rc,2);"%":
PRINT:PRINT"DER BUCHSTABE         C"
1650 IF r=rt THEN PRINT ROUND(rt,2);"%":
PRINT:PRINT"DER BUCHSTABE         T"
1660 PEN 3:LOCATE 8,23:PRINT"NOCH EIN MU
STER BEARBEITEN"
1670 LOCATE 7,25:PRINT"BITTE DIE TASTE <
S> DRUECKEN"
1680 IF (INKEY$="s") OR (INKEY$="S") THE
N CLEAR:kh=1:GOTO 10
```

Programm
ACTSYM.MER

```
1690 GOTO 1680
2000 PEN 3:PRINT"ZU WENIGE DATEN":PRINT:
PRINT"DIE EINGABE":PRINT:PRINT"KANN NICH
T IDENTIFIZIERT WERDEN":PRINT
2010 PEN 2:PRINT"SIE DECKT NUR ETWA":PRI
NT:PRINT ROUND(j*100,2);"%":PRINT:PRINT"
EINES ERKENNBARERN MUSTERS AB":PRINT
2020 PRINT"BITTE GEBEN SIE":PRINT:PRINT"
EIN GROESSERES MUSTER EIN"
2030 RETURN
10000 REM > Bildschirmgestaltung <
10010 MODE 1:BORDER 0:INK 0,0:INK 1,24:I
NK 2,20:INK 3,6
10020 DIM g(27,7):RANDOMIZE TIME
10030 FOR j=0 TO 7:READ a(j),c(j),t(j):N
EXT j
10040 IF kh=1 GOTO 11000
10050 PEN 1:FOR j=1 TO 20:FOR i=1 TO 40:
LOCATE i,j:PRINT CHR$(233):NEXT i:NEXT j
10100 FOR j=0 TO 7
10110 g(0,j)=a(j) AND 128:g(10,j)=c(j) A
ND 128:g(20,j)=t(j) AND 128
10120 g(1,j)=a(j) AND 64:g(11,j)=c(j) AN
D 64:g(21,j)=t(j) AND 64
10130 g(2,j)=a(j) AND 32:g(12,j)=c(j) AN
D 32:g(22,j)=t(j) AND 32
10140 g(3,j)=a(j) AND 16:g(13,j)=c(j) AN
D 16:g(23,j)=t(j) AND 16
10150 g(4,j)=a(j) AND 8:g(14,j)=c(j) AND
 8:g(24,j)=t(j) AND 8
10160 g(5,j)=a(j) AND 4:g(15,j)=c(j) AND
 4:g(25,j)=t(j) AND 4
10170 g(6,j)=a(j) AND 2:g(16,j)=c(j) AND
 2:g(26,j)=t(j) AND 2
10180 g(7,j)=a(j) AND 1:g(17,j)=c(j) AND
 1:g(27,j)=t(j) AND 1
10190 NEXT j
10200 x=INT(RND(1)*28):y=INT(RND(1)*8)
10210 IF g(x,y)>0 THEN PEN 3:LOCATE x+7,
y+4:PRINT CHR$(233)
10220 z=z+1:IF z<500 GOTO 10200
10230 PEN 3:FOR i=0 TO 29:LOCATE i+6,3:P
RINT CHR$(233):NEXT i
10240 FOR i=0 TO 7:LOCATE 6,i+4:PRINT CH
R$(233):LOCATE 35,i+4:PRINT CHR$(233):NE
XT i
10250 FOR i=0 TO 29:LOCATE i+6,12:PRINT
CHR$(233):NEXT i
10300 FOR i=0 TO 27
10310 FOR j=0 TO 7
10320 IF g(i,j)=0 THEN PEN 3
10330 IF g(i,j)>0 THEN PEN 2
10340 LOCATE i+7,j+4:PRINT CHR$(233)
10350 NEXT j
10360 NEXT i
10400 FOR j=1 TO 30
10410 g$=MID$("* MUSTER-ERKENNUNGS-PROGR
AMM *",j,1)
10420 LOCATE 5+j,16:PRINT g$
10430 FOR i=0 TO 250:NEXT i
```

Programm
ACTSYM.MER

```
10440 NEXT j
10500 PEN 2:LOCATE 3,25:PRINT"< BITTE IR
GENDEINE TASTE DRUECKEN >"
10510 IF INKEY$="" GOTO 10510
11000 MODE 1:INK 1,0:INK 2,25
11010 WINDOW 16,27,4,15:WINDOW #1,1,40,2
0,25
11020 PEN 2:FOR j=0 TO 7:FOR i=0 TO 7:LO
CATE i+2,j+2:PRINT CHR$(143):NEXT i:NEXT
 j
11030 PEN 3:LOCATE 2,1:PRINT CHR$(241):L
OCATE 2,10:PRINT CHR$(240):LOCATE 1,2:PR
INT CHR$(243):LOCATE 10,2:PRINT CHR$(242
)
11040 PAPER #1,0:PEN #1,3:PRINT #1,"
    BEWEGEN MIT DEM JOYSTICK"
11050 PRINT #1:PRINT #1,"SCHREIBEN / LOE
SCHEN MIT DEM FEUERKNOPF"
11060 PRINT #1:PRINT #1,"   ZEICHEN FERT
IG: TASTE <E> DRUECKEN";
20000 REM > Sonderzeichen <
20010 SYMBOL AFTER 250
20020 SYMBOL 250,255,195,129,129,129,129
,195,255
20030 RETURN
30000 DATA 24,60,255,60,126,255,60,228,2
4,102,192,24,126,192,24,255,228,24,195,1
26,24,195,60,24
```

Programmbeschreibung
ACTSYM.MER

Zeile 10 veranlaßt den Sprung in ein Unterprogramm, das eine aufwendige Titelanimation enthält. Wird solches Augenpulver bemüht, folgt in der Praxis oft ein schlechtes Programm. Der Aufwand wurde in diesem Fall betrieben, um bildlich zu demonstrieren, wie sich einzelne Punkte, hier zufällig verteilt, in die vorgegebenen Masken fügen. Die gezeigten Buchstaben »A«, »C« und »I« haben die Form der verwendeten Masken. Ab **Zeile 10500** wird dann der Bildschirm für das eigentliche Programm gestaltet.
In **Zeile 100** wird abgefragt, ob die Taste <E> gedrückt wurde. Die **Zeilen 100 bis 160** nehmen die Joystickbewegungen auf; die **Zeilen 170 und 180** bewegen die Pfeile (Cursor) zur jeweils gegenüberliegenden Seite, wenn der Rand des Schreibfensters überschritten wurde.
Zeile 200 löscht die Position der Pfeile, die an den Rändern des Schreibfeldes laufen, wenn der Joystick bewegt wird. Die Pfeile markieren mit ihrem Schnittpunkt die Stelle, die gelöscht oder beschrieben werden kann. Dann schreibt **Zeile 220** die Pfeile an die neue Position. In **Zeile 230** wird der Feuerknopf abgefragt. Wurde er gedrückt, wird die Variable p auf 1 gesetzt.
In der Feldvariablen m wird für jeden der 8 x 8 Bildpunkte ein Wert bewahrt, der 0 ist, wenn kein Punkt, und

Programmbeschreibung
ACTSYM.MER

1 ist, wenn ein Punkt gesetzt ist. Der Inhalt des Variablenelements, m(x—2,y—2), entspricht der von den Pfeilen angezeigten Bildschirmposition. Er wird durch den Operator XOR umgedreht. 1 wird zu 0 und umgekehrt. Anhand des neuen Wertes von m(x—2,y—2) wird an die entsprechende Bildschirmposition (**Zeile 250**) der Hintergrund (**Zeile 260**) oder ein Punkt (**Zeile 270**) geschrieben.
Die Warteschleife in **Zeile 280** bremst den Eifer des Computers, der die Joystickeingabe viel schneller umsetzt, als der Anwender reagieren kann, bevor in **Zeile 290** durch Rücksprung nach **Zeile 100** wieder an den Spieler übergeben wird.
Wenn in **Zeile 100** die Auswertung aufgerufen wird, fährt das Programm bei **Zeilennummer 1000** fort. Weil eine gewisse Rechenzeit benötigt werden kann, wird zuerst eine entsprechende Meldung ausgegeben.
In den **Zeilen 1100 bis 1120** wird die Anzahl der eingegebenen Punkte gezählt und in der Variablen pn notiert; in den **Zeilen 1130 bis 1220** wird dann das in m(i,j) gespeicherte Punktmuster umgerechnet, das der Anwender in das Bildschirmfenster geschrieben hat. Die acht Punkte einer Zeile werden ihrer Position entsprechend nach dem Prinzip der Bit-Muster zu einem 8-Bit-Wert zusammengefaßt. Die so gewonnenen 8 Bytes werden in der Variablen n(j) verwahrt.
In den **Zeilen 1300 bis 1340** werden die acht Bytes des eingegebenen Musters mit den acht Bytes der Masken verglichen. Das Programm kennt nur die drei in a(j), c(j) und t(j) erfaßten Muster. Der Operator AND vergleicht je zwei Bytes bitweise und resultiert nur übereinstimmende Bits. Diese übereinstimmenden Bits werden (zu einem Byte zusammengefaßt) an die Hilfsvariablen qa(j), qc(j) und qt(j) übergeben.
Für jede der berechneten Hilfsvariablen wird dann ermittelt, wie viele gesetzte Punkte sie beschreibt, das heißt, wie viele Punkte der Eingabe mit Punkten der Maske übereinstimmen. Die gefundene Anzahl wird an die Variable pa, pc bzw. pt übergeben.
Zeile 1500 definiert eine Funktion, welche die gefundene Punkteübereinstimmung in Prozent umrechnet. In den **Zeilen 1510 bis 1530** wird diese Funktion durch Einsetzen verschiedener Werte angewendet. Das Resultat (ra, rc oder rt) entsteht, indem die Anzahl der übereinstimmenden Punkte durch die Anzahl der Punkte der jeweiligen Maske geteilt, mit 100 multipliziert und durch die Anzahl der eingegebenen Punkte geteilt wird.
Die Variablen ja, jc und jt halten den Quotienten aus der Anzahl der (mit der jeweiligen Maske) übereinstimmen-

Programmbeschreibung
ACTSYM.MER

den Punkte und der Anzahl der Punkte der Maske. In **Zeile 1550** findet MAX den größten dieser Werte und übergibt ihn an j.
Wenn die größte Übereinstimmung j kleiner als 33% ist, wird das Unterprogramm ab **Zeile 2000** aufgerufen. Von dort erfolgt die Meldung, daß die Eingabe nicht hinreichend identifiziert werden kann. Danach wird nach **Zeile 1660** verzweigt, wo das Programm mit der Abfrage für einen neuen Durchlauf aufgefangen wird.
Durch die Überprüfung des Quotienten j wird verhindert, daß bei Eingabe nur eines einzigen Punktes, der zufällig mit einem Punkt aus dem Muster einer Maske übereinstimmt, ein Zeichen (100%ig) erkannt wird. Die vorgegebenen 33% fordern also, daß so viele Punkte vom Anwender gesetzt wurden, daß mindestens ein Drittel von ihnen innerhalb der Maske liegt.
Wurde der j-Wert akzeptiert, wird in der Variablen r der größte Wert aus ra, rc und rt erfaßt. Ist die nach der oben erklärten Formel größte Übereinstimmung r kleiner als 75%, wird das Zeichen als ebenfalls nicht identifizierbar eingestuft. Sonst wird der mit der Maske assoziierte Buchstabe und der Prozentsatz der Übereinstimmung ausgegeben.
Das Programm kann theoretisch beliebig erweitert werden. Sie müssen nur die Masken für weitere Buchstaben entwerfen und als Bitmuster-Bytes in Datazeilen bei **Zeilennummer 30000** ablegen. Überall, wo für einzelne Masken Werte zu berechnen sind, müssen entsprechend mehr Arbeitsschritte für die neuen Masken eingefügt werden. Die kritische Akzeptanz, wann ein Buchstabe als identifiziert gelten soll, und die Unterscheidung zwischen ähnlichen Buchstaben wird natürlich immer diffiziler. Die in diesem Programmbeispiel verwendeten Buchstaben »A«, »C« und »T« sind in ihrer Form signifikant voneinander verschieden. Die Schwelle so zu justieren, daß auch zwischen »L«, »F« und »E« getrennt werden kann, wird viel Fingerspitzengefühl verlangen. Eine Unterscheidung von »O« und »Q« oder »I« und »J« wird mit dieser einfachen Programmstruktur wohl kaum zu realisieren sein.

Der Pfadfinder

Lernen, beim Wort genommen, gehört wie *Wissen* (*list*) zur Wortgruppe *leisten*. Es geht auf die gotische Wurzel *leis* (Spur, Bahn, Furche) zurück. Die Bedeutung des Wortes *Wissen* hat sich aus *nachgespürt haben*, *Lernen* aus *wissend werden* entwickelt.

Im Rückblick auf die Geschichte unserer Sprache haben unsere Vorfahren mit der Tätigkeit des Lernens also die Vorstellung verbunden, daß man wissend wird, indem man eine Spur verfolgt, wozu in der Regel auch das Suchen gehört. Die Psychologen haben für diesen Vorgang den Begriff des Lernens am Erfolg geprägt, des Versuchs und Irrtums (*trial and error*). Wissend werden heißt aber auch, die gefundene Spur zu erinnern.

Labyrinthe:
Nicht nur ein Spaß, sondern auch Objekte der Intelligenz-Forschung

Dieser Exkurs mag erhellen, warum Lernpsychologen so gerne mit Labyrinthen arbeiten. Eine hungrige Ratte wird in einem Irrgarten ausgesetzt. Am Ausgang erwartet sie eine Belohnung in Form von Nahrung. Wenn die Ratte bei wiederholten Versuchen den Weg immer leichter findet, Umwege und Sackgassen vermeidet, hat sie unzweifelhaft gelernt.

Welche Rückschlüsse daraus auf die Intelligenz der Ratte gezogen werden können, sei einmal dahingestellt. Zwar ist Lernen eine intelligente Tätigkeit. Aber kann aus dem Vorgang solchen Lernens eine Intelligenz abgeleitet werden, die auch die Fähigkeit des Verstehens einschließt?

Die Ratte ist durch Märchen und Sagen sowie als Objekt wissenschaftlicher Forschung zu einem Tier mit einem starken Fluidum geworden. Als besonders anpassungsfähig ist sie heute zudem fast weltweit vertreten. Sie soll auch in diesem Buch nicht übergangen werden. In einem Labyrinth soll sie den kürzesten Weg zum Ziel finden.

Eine für unsere Zwecke geeignete Computerratte ist schnell aus ein paar Bildpunkten zusammengesetzt. Aber woher nehmen wir ein Labyrinth? Aus Unmengen grafischer Daten zusammenbauen wollen wir es jedenfalls nicht. Dazu ist nicht nur unser Forschungsetat zu klein, dazu sind wir auch zu faul (oder?). Außerdem würden wir auf diese mühselige Weise nur ein einziges, immer gleiches Labyrinth auf die Mattscheibe zeichnen. Ein Algorithmus muß her, der den Rechner in die Lage versetzt, selbständig beliebig neue Labyrinthe hervorzubringen.

Definition des Irrgartens

Wir definieren den Irrgarten als begrenzte geometrische Fläche, die so in Wände und Gänge eingegliedert ist, daß von einem Eintrittspunkt in die Fläche ein Weg zu einem Austrittspunkt führt. Rationales Prinzip des Irrgartens ist also, daß Ein- und Ausgang niemals durch eine durchgehende Wand getrennt werden. Anders gesagt: Jede zusammenhängende Konfiguration von Wandteilen darf an höchstens einer Stelle mit der Umlaufkante der Fläche verbunden sein. Jede Wandkonfiguration, die zwei und mehr Verbindungspunkte zur Umlaufkante hat, zerteilt die Fläche in zwei oder mehr in sich geschlossene Teile. Befinden sich dann Ein- und Ausgang in verschiedenen Teilen, kann der Irrgarten nicht durchquert werden. Andernfalls zerfällt die Fläche in einen kleineren Teil, der durchquert werden kann, und in einen oder mehrere andere nicht betretbare.

Das Programm muß demnach angewiesen werden, niemals geschlossene Wandverbindungen von Umlaufkante zu Umlaufkante herzustellen. Zur Vereinfachung gehen wir von einem Rechteck aus, das in Quadrate gerastert ist. Auf dem Monitor lassen wir 25 x 25 Schreibstellen bearbeiten. Jedes Rasterelement kann entweder mit Wand oder Gang gefüllt werden. Alle Wände und Gänge stehen immer nur senkrecht aufeinander und sind immer ein Rasterelement breit. Die Begrenzung der Fläche besteht aus einer umlaufenden Wand mit zwei Öffnungen, die ebenfalls diesen Bauvorschriften unterliegen.

Der im Programm LABY.GRF verwendete Algorithmus kann etwa so in Worte gefaßt werden:

Ziehe zuerst die umgrenzende Mauer um die Fläche. Öffne den Eingang an oberster Position der linken Wand und einen Ausgang an zufälliger Stelle in der rechten Wand. Suche dann wiederholt einen zufälligen Rasterpunkt, auf dem eine Wand stehen darf und der keine bereits stehende Wand berührt. Ziehe von diesem Punkt eine Wand in zufällig gewählter Richtung, bis die Verbindung mit einer anderen Wand hergestellt ist. Damit ist gewährleistet, daß alle gezogenen Wandstücke nur an einem Punkt mit einem anderen Wandstück oder der umlaufenden Wand zusammenhängen, also nie eine zwei Wandstücke verbindende Wand gezogen wird. Folglich muß ein Durchgang von Start nach Ziel offen bleiben. Abschließend ist noch zu überprüfen, daß bei der zufälligen Verteilung von Wandstücken keine Baulücke geblieben ist, um die Regel zu erfüllen, daß Wände wie Gänge immer nur ein Rasterelement (eine Schreibstelle auf dem Monitor) breit sind.

Programm
LABY.GRF

```
1 REM LABY.GRF
10 RANDOMIZE TIME
20 GOSUB 10010
30 END
10000 REM > Umgrenzung <
10010 BORDER 0:INK 0,0:INK 1,2:INK 2,18:
INK 3,26:PAPER 0:CLS
10020 DIM l(24,24):FOR j=0 TO 24:FOR i=0
 TO 24:l(i,j)=16:NEXT i:NEXT j
10030 PEN 2:FOR j=0 TO 24:LOCATE 16+j,1:
PRINT CHR$(207);:LOCATE 16+j,25:PRINT CH
R$(207);:l(j,0)=0:l(j,24)=0:NEXT j
10040 FOR j=2 TO 23:LOCATE 40,j:PRINT CH
R$(207);:LOCATE 16,26-j:PRINT CHR$(207);
:l(24,j-1)=0:l(0,25-j)=0:NEXT j
10050 LOCATE 40,24:PRINT CHR$(207):l(24,
23)=0
10060 y=INT(RND(1)*12)*2+2
10070 LOCATE 40,y:PRINT CHR$(32):l(24,y-
1)=32
10100 REM > zufällig füllen <
10110 le=7
10130 z=z+1:IF z>30 GOTO 10210
10140 x=INT(RND(1)*5)*4+18
10150 y=INT(RND(1)*5)*4+3
10160 GOSUB 11010
10170 GOTO 10130
10200 REM > Lücken absuchen <
10210 le=3
10220 FOR j=0 TO 10
10230 FOR i=0 TO 10
10240 x=18+2*j
10250 y=3+2*i
10260 GOSUB 11010
10270 NEXT i
10280 NEXT j
11000 REM > Wände errichten <
11010 IF l(x-16,y-1)=0 THEN RETURN
11020 LOCATE x,y:PRINT CHR$(207):l(x-16,
y-1)=0
11030 r=INT(RND(1)*4)
11040 IF r=0 THEN v=0:w=1
11050 IF r=1 THEN v=0:w=-1
11060 IF r=2 THEN v=1:w=0
11070 IF r=3 THEN v=-1:w=0
11100 st=0
11110 FOR jj=0 TO le
11120 IF st=1 GOTO 11160
11130 x=x+v:y=y+w
11140 IF l(x-16,y-1)=0 THEN st=1:GOTO 11
160
11150 LOCATE x,y:PRINT CHR$(207):l(x-16,
y-1)=0
11160 NEXT jj
11170 RETURN
```

Programmbeschreibung LABY.GRF

Die zufällige Labyrintherzeugung ist als Unterprogramm geschrieben, das bei **Zeile 10000** beginnt. Zuerst werden Farbwerte bestimmt.
Die gefundene Gestaltung soll in Form von Daten notiert werden. Dazu dimensioniert **Zeile 10010** die Feldvariable l mit 25 x 25 Elementen, von denen sich jedes auf eine entsprechende Schreibstelle des Bildschirms bezieht. Alle 625 Elemente der Variablen werden mit dem Wert 16 gefüllt.
Die **Zeilen 10030 bis 10070** errichten die umlaufende Wand und öffnen den definierten Ein- und den zufälligen Ausgang. In der Feldvariablen l werden Wände durch eine 0 dargestellt. Der Eingang behält in der Feldvariablen l den Wert 16; der Ausgang wird mit 32 geschmückt.
Ab **Zeile 10100** werden in der beschriebenen Weise zufällige Wandstücke eingetragen. Da anfangs die Fläche leer ist, könnten einige sehr lange Wandteile entstehen. Deshalb wird durch die Variable le die Länge auf maximal acht Schreibstellen beschränkt. Es wird mit Null beginnend gezählt.
Zeile 10130 überwacht mit z, daß nur 30 Versuche unternommen werden, zufällige Wandstücke auf der Fläche zu plazieren. Dann finden die **Zeilen 10130 und 10140** zufällige Werte für x und y, die als Koordinaten weiterverarbeitet werden, wenn in **Zeilennummer 10160** das Unterprogramm ab **Zeile 11010** aufgerufen wird, wo die eigentliche Darstellung des Labyrinths erfolgt.
Sind die 30 Durchläufe abgearbeitet, wird das Programm bei **Zeile 10210** fortgesetzt, wo die maximale Wandlänge le auf den Wert 3 (vier Elemente) gesetzt wird. Die doppelte FOR-NEXT-Schleife von **Zeile 10220 bis 10280** sucht schließlich das gesamte Labyrinth ab, um etwaige Baulücken zu entdecken. Auch aus dieser Schleife heraus wird das Unterprogramm bei **Zeile 11010** aufgerufen.
Bevor weitergehende Baumaßnahmen eingeleitet werden, überprüft **Zeile 11010**, ob die gefundene Stelle noch unbebaut ist. Nur wenn das nicht der Fall ist, beschreibt **Zeile 11020** die durch (x,y) bestimmte Position des Bildschirms und das entsprechende Element der Feldvariablen l.
Die **Zeilen 11030 bis 11070** setzen eine zufällig gefundene Zahl in Werte v und w um, über welche die Baurichtung der Wand kontrolliert wird.
Der Bau erfolgt in den **Zeilen 11100 bis 11160**. Zuerst wird die Flag-Variable st auf 0 gesetzt. Die FOR-NEXT-Schleife arbeitet in jj die Werte 0 bis le durch, womit die maximale Wandlänge in Rasterpunkten bestimmt ist.

Zeile 11130 verändert die Koordinaten unter Verwendung von v und w; **Zeile 11140** überprüft, ob der geplante Bauabschnitt an eine bestehende Wand stößt. Ist das der Fall, wird das Flag st auf 1 gesetzt und damit für die restliche Bearbeitung der Schleife in **Zeile 11120** ein Baustop verhängt. Andernfalls werden in **Zeile 11150** die Bildschirmposition und die Feldvariable l entsprechend beschrieben.
Tatsächlich entsteht auf dem Bildschirm relativ flink, und immer wieder anders strukturiert, ein Irrgarten, durch den ein Lösungsweg hindurchführt. Doch erzeugt der verwendete Algorithmus ein Gänge/Wege-Muster, das Kreisverkehr zuläßt: Mehrere Wege führen zum Ziel. Unsere Ratte könnte endlos umherirren wie um die Häuserblocks einer Großstadt, um vielleicht irgendeinmal zufällig den Ausgang zu erheischen, wenn wir sie nicht mit der Fähigkeit ausstatten, solche Rückschritte zu erkennen. Da es aber in diesem Beispiel um das Lernen, also das Finden eines (Lösungs-) Weges nach Versuch und Irrtum geht, soll das Testgelände für unseren tapferen Vorreiter der Wissenschaft durch einen anderen, ähnlichen Labyrinth-Algorithmus gestaltet werden, durch den ein — und nur ein — Weg zwischen Ein- und Ausgang gelegt wird.
Alternative Wege durch den Irrgarten entstehen, wenn Wandkonfigurationen wie Inseln im Innern der Fläche liegen, also keine Verbindung zur Umlaufkante haben. Sind alle Wandteile mit der Kante verbunden, gibt es nur zwei große, ineinander verzahnte Wandkomplexe.

Programm LABYRATT.LRN

```
1 REM LABYRATT.LRN
10 RANDOMIZE TIME
20 GOSUB 60010
30 GOSUB 10010
40 GOSUB 15010
90 WINDOW #1,1,15,5,25
100 rn=INT(RND(1)*4)
110 IF rn=(r+2) MOD 4 GOTO 100
120 IF rn=0 THEN x=0:y=1
130 IF rn=1 THEN x=1:y=0
140 IF rn=2 THEN x=0:y=-1
150 IF rn=3 THEN x=-1:y=0
160 IF l(rx+x-16,ry+y-1)=2 GOTO 100
170 IF rm(rx+x-16,ry+y-1)=1 THEN r=(r+2)
 MOD 4:GOTO 100
180 IF rm(rx+x-16,ry+y-1)=2 GOTO 100
200 LOCATE rx+x,ry+y:PRINT CHR$(166)
210 FOR j=0 TO 100:NEXT j
220 LOCATE rx,ry:PRINT" "
230 rx=rx+x:ry=ry+y
240 IF l(rx-16,ry-1)=-1 THEN GOTO 20010
250 LOCATE rx+x,ry+y:PRINT CHR$(166)
```

Programm
LABYRATT.LRN

```
260 FOR j=0 TO 100:NEXT j
270 LOCATE rx,ry:PRINT" "
280 rx=rx+x:ry=ry+y
290 s=s+1:sz=sz+1:IF s>99 THEN s=0
300 r=rn
310 IF ww=1 THEN w(s)=rn
320 GOSUB 16010
330 IF f=1 AND e=1 THEN rm(rx-x-16,ry-y-
1)=2:f=0:p=p+1
340 e=0
350 GOTO 100
10000 REM > Umgrenzung <
10010 BORDER 0:INK 0,0:INK 1,11:INK 2,18
:INK 3,26:PAPER 0:CLS
10020 DIM l(24,24):LOCATE 2,2:PEN 3:PRIN
T CHR$(166)
10030 PEN 2:FOR j=0 TO 24:LOCATE 16+j,1:
PRINT CHR$(207);:LOCATE 16+j,25:PRINT CH
R$(207);:l(j,0)=2:l(j,24)=2:NEXT j
10040 FOR j=2 TO 23:LOCATE 40,j:PRINT CH
R$(207);:LOCATE 16,26-j:PRINT CHR$(207);
:l(24,j-1)=2:l(0,25-j)=2:NEXT j
10050 LOCATE 40,24:PRINT CHR$(207):l(24,
23)=2
10060 y=INT(RND(1)*12)*2+2
10070 LOCATE 40,y:PEN 1:PRINT"?":PEN 2:l
(24,y-1)=-1
10080 LOCATE 40,25:PRINT" ";:LOCATE 40,2
5:PRINT CHR$(167);:PEN 2
11000 REM > zufallig fullen <
11010 z=z+1:IF z>700 GOTO 12010
11020 x=INT(RND(1)*13)*2+16
11030 y=INT(RND(1)*13)*2+1
11040 GOSUB 13010
11050 GOTO 11010
12000 REM > Lücken absuchen <
12010 FOR j=0 TO 12
12020 FOR i=0 TO 12
12030 x=16+2*j
12040 y=1+2*i
12050 IF l(2*j,2*i)=0 THEN GOSUB 14010
12060 NEXT i
12070 NEXT j
12080 RETURN
13000 REM > Wände errichten <
13010 IF l(x-16,y-1)=0 THEN RETURN
13020 zz=0:a=INT(RND(1)*4)*2+2
13030 GOSUB 17010
13040 x=x+v:y=y+w
13050 IF x+v<16 OR x+v>40 THEN RETURN
13060 IF y+w<1 OR y+w>25 THEN RETURN
13070 IF l(x-16+v,y-1+w)=2 THEN RETURN
13080 LOCATE x,y:PRINT CHR$(207):l(x-16,
y-1)=2
13090 zz=zz+1:IF zz<a GOTO 13040
13100 RETURN
14000 REM > Lücken füllen <
14010 LOCATE x,y: PRINT CHR$(207):l(x-16
,y-1)=2
14020 GOSUB 17010
```

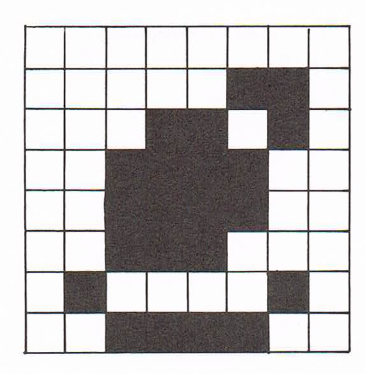
Programm
LABYRATT.LRN

```
14030 x=x+v:y=y+w
14040 IF l(x-16,y-1)=2 THEN RETURN
14050 LOCATE x,y:PRINT CHR$(207):l(x-16,
y-1)=2
14060 GOTO 14030
15000 REM > Ratte ins Lab <
15010 PEN 3:FOR i=3 TO 17
15020 LOCATE i,2:PRINT CHR$(166)
15030 FOR j=0 TO 50:NEXT j
15040 LOCATE i-1,2:PRINT" "
15050 FOR j=0 TO 50:NEXT j
15060 NEXT i
15070 DIM rm(24,24),w(99)
15080 s=0:sz=0:r=1:w(0)=1:rm(0,1)=1::rx=
17:ry=2
15090 RETURN
16000 REM > Orientierung <
16010 g=l(rx-16,ry)+l(rx-15,ry-1)+l(rx-1
6,ry-2)+l(rx-17,ry-1)+rm(rx-16,ry)+rm(rx
-15,ry-1)+rm(rx-16,ry-2)+rm(rx-17,ry-1)
16020 IF g=6 THEN f=1:r=(r+2) MOD 4
16030 IF g<4 THEN e=1
16040 RETURN
17000 REM > zufällige Richtung <
17010 r=INT(RND(1)*4)
17020 IF r=0 THEN v=0:w=1
17030 IF r=1 THEN v=1:w=0
17040 IF r=2 THEN v=0:w=-1
17050 IF r=3 THEN v=-1:w=0
17060 RETURN
20000 REM > Ausgang erreicht <
20010 ww=1:IF p=0 GOTO 20230
20020 PRINT #1,"Ich bin da!"
20030 PRINT #1,"Diesmal habe"
20040 PRINT #1,"ich";sz
20050 PRINT #1,"Schritte"
20060 PRINT #1,"gebraucht."
20070 PRINT #1,"Nächstes Mal"
20080 PRINT #1,"bin ich"
20090 PRINT #1,"schneller!"
20100 PRINT #1:PRINT #1,"Probier's:"
20110 PRINT #1,"Drück eine"
20120 PRINT #1,"Taste."
20130 LOCATE rx,ry:PEN 1:PRINT"!"
20140 FOR j=0 TO 500:NEXT j
20150 LOCATE rx,ry:PEN 3:PRINT CHR$(166)
20160 FOR j=0 TO 500:NEXT j
20170 IF INKEY$="" GOTO 20130
20180 LOCATE rx,ry:PEN 1:PRINT"?"
20190 p=0:s=0:sz=0:r=1:rx=17:ry=2
20200 PEN 3:CLS #1
20210 GOTO 90
20220 REM > kürzester Weg <
20230 PRINT #1,"Jetzt kenne"
20240 PRINT #1,"ich den"
20250 PRINT #1,"kürzesten Weg."
20260 PRINT #1,"Er ist"
20270 PRINT #1,s;"Schritte"
20280 PRINT #1,"lang."
20290 PRINT #1,"Ich finde ihn"
```

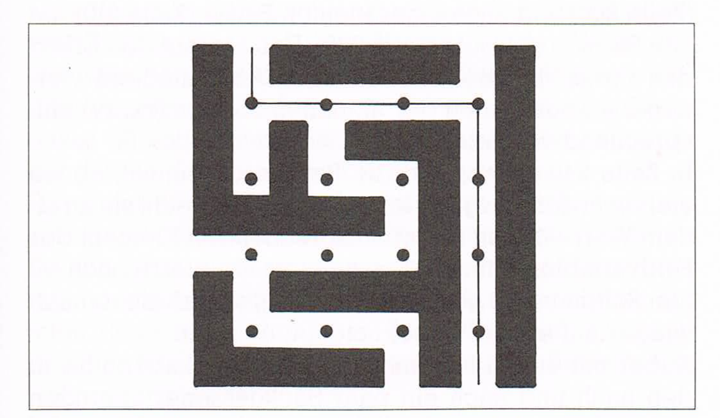
Programm
LABYRATT.LRN

```
20300 PRINT #1,"sofort wieder."
20310 PRINT #1:PRINT #1,"Probier's:"
20320 PRINT #1,"Drück eine"
20330 PRINT #1,"Taste."
20340 LOCATE rx,ry:PEN 1:PRINT"!"
20350 FOR j=0 TO 300:NEXT j
20360 LOCATE rx,ry:PEN 3:PRINT CHR$(166)
20370 FOR j=0 TO 300:NEXT j
20380 IF INKEY$="" GOTO 20340
20390 LOCATE rx,ry:PEN 1:PRINT"?"
20400 r=1:rx=17:ry=2
20410 WINDOW #1,1,15,5,25:PEN 3:CLS #1
20420 FOR i=1 TO s
20430 IF w(i)=0 THEN x=0:y=1
20440 IF w(i)=1 THEN x=1:y=0
20450 IF w(i)=2 THEN x=0:y=-1
20460 IF w(i)=3 THEN x=-1:y=0
20470 LOCATE rx+x,ry+y:PRINT CHR$(166)
20480 FOR j=0 TO 100:NEXT j
20490 LOCATE rx,ry:PRINT" "
20500 rx=rx+x:ry=ry+y
20510 LOCATE rx+x,ry+y:PRINT CHR$(166)
20520 FOR j=0 TO 100:NEXT j
20530 LOCATE rx,ry:PRINT" "
20540 rx=rx+x:ry=ry+y
20550 NEXT i
20560 PRINT #1,"Das war's."
20570 PRINT #1,"Noch mal?":PRINT #1
20580 PRINT #1,"Druck eine"
20590 PRINT #1,"Taste."
20600 LOCATE rx,ry:PRINT" "
20610 LOCATE rx+1,ry:PEN 1:PRINT"!"
20620 FOR j=0 TO 250:NEXT j
20630 LOCATE rx+1,ry:PEN 3:PRINT CHR$(16
6)
20640 FOR j=0 TO 250:NEXT j
20650 IF INKEY$="" GOTO 20610
20660 LOCATE rx+1,ry:PEN 1:PRINT"?"
20670 r=1:rx=17:ry=2
20680 GOTO 20410
60000 REM > Sonderzeichen <
60010 SYMBOL AFTER 91
60020 SYMBOL 91,66,24,60,102,126,102,102
,0
60030 SYMBOL 92,130,56,108,198,198,108,5
6,0
60040 SYMBOL 93,66,0,102,102,102,102,60,
0
60050 SYMBOL 123,68,0,120,12,124,204,118
,0
60060 SYMBOL 124,36,0,60,102,102,102,60,
0
60070 SYMBOL 125,36,0,102,102,102,102,62
,0
60080 SYMBOL 126,60,102,124,102,102,102,
108,96
60090 SYMBOL 166,0,6,26,60,60,56,65,60
60100 SYMBOL 167,196,204,216,243,243,216
,204,196
60110 RETURN
```

Im Unterprogramm ab **Zeile 60000** werden nicht nur Zeichen als deutsche Umlaute, sondern auch ASCII 166 als Ratte umdefiniert. Das Unterprogramm ab **Zeilennummer 10000** erzeugt einen zufälligen Irrgarten. In dieser Version wird aber nicht ein beliebiger Punkt auf der Fläche gesucht, von dem dann ein Wandteil gezogen wird, das Programm sucht einen Punkt auf einer schon bestehenden Wand und errichtet von dort ausgehend ein neues Stück. Das Labyrinth wächst also von den Umlaufkanten in die anfangs leere Fläche hinein. Damit ist gewährleistet, daß die Wände, wie oben erläutert, zusammenhängen.
Das dritte Unterprogramm ab **Programmzeile 15000** läßt die Ratte vom linken, nicht vom Irrgarten bedeckten Teil des Bildschirms zum Eingang laufen.
Die Konstruktionsvorschrift für den Irrgarten bestimmt, daß Wände wie Gänge immer genau eine Schreibstelle breit sein müssen. Für den Weg der Ratte sind deshalb nur die jeweils zwei Schreibstellen voneinander entfernten Punkte wichtig, an denen sich Gänge verzweigen können. Diese Punkte sind in der Abbildung markiert. Sie sind grundsätzlich Teil eines Ganges, während die Schreibstellen dazwischen je nach Gestalt des Labyrinths mit Wand oder Gang belegt sind.



Die Ratte wird durch das umdefinierte Zeichen mit dem ASCII-CODE 166 dargestellt.



Aufgrund der Konstruktionsvorschrift für das Labyrinth erfordern nur die mit einem Punkt markierten Positionen eine Entscheidung vom Programm, das die Ratte steuert.

Bei **Zeile 100** beginnt die Ratte mit der Suche. Sie streckt die Nase schnuppernd in die Luft und wählt zufällig eine Richtung aus, in die sie laufen kann. Zuerst wird überprüft, ob die neue Richtung rn nicht genau der alten Richtung, die in r notiert ist, entgegenläuft, denn das würde bedeuten, daß die Ratte sich in die Richtung

Programmbeschreibung
LABYRATT.LRN

wendet, aus der sie gekommen ist. Wenn das nicht der Fall ist, setzen die **Zeilen 120 bis 150** die gewählte Richtung in die Bewegungskoordinaten x und y um.
In **Zeile 160** schaut die Ratte nach, ob die Schreibstelle, auf die sie x und y führen würden, mit einer Mauer bebaut ist. Die Variablen rx und ry beschreiben als Koordinaten die aktuelle Position der Ratte.
So wie ein Bauplan des gesamten Irrgartens in der Feldvariablen l notiert ist, so steht für die Ratte ein Gedächtnis in Form einer Feldvariablen rm zur Verfügung, die genauso viele Elemente enthält. In rm merkt sich die Ratte, was sie über das Labyrinth herausgefunden hat und für ihre Suche wichtig ist. Damit das Tier den Versuchsaufbau nicht fluchtartig durch den Eingang wieder verläßt, wurde bereits in **Zeile 15080** ein Eingriff in ihr Gedächtnis vorgenommen und der Eingang mit einer 1 zum Tabu erklärt. Hier in **Zeile 170** wird die Fluchtblockade überprüft und gegebenenfalls die Marschrichtung umgedreht.
Schließlich erinnert sich die Ratte noch, ob sie den eingeschlagenen Weg vielleicht schon einmal gegangen ist und dabei festgestellt hat, daß er in einer Sackgasse endet.
Erst wenn alle diese Befunde negativ sind, macht die Ratte den geplanten Schritt. Dabei wird zuerst ein Rattensymbol an die neue Bildschirmstelle geschrieben (**Zeile 200**) und nach einer kleinen Pause (**Zeile 210**) die alte Position gelöscht (**Zeile 220**). Das erzeugt den Effekt des etwas »kribbeligen« Laufens. Abschließend werden die Koordinaten der aktuellen Position (rx, ry) entsprechend verändert.
In **Zeile 240** schnuppert die Ratte noch einmal, ob sie vielleicht den Ausgang erreicht hat. Das riecht sie an einem Wert —1, den sie im entsprechenden Element der Feldvariablen l findet. Sonst macht sie gleich noch einen Schritt in die gewählte Richtung, so daß sie danach wieder auf einem Wegeknotenpunkt steht.
Außer mit einem lückenhaften Plan des Labyrinths, in den nach und nach ein paar Sackgassenerfahrungen eingetragen werden, ist die Ratte noch mit einem anderen Gedächtnisteil ausgestattet. Hier notiert sie die Richtung jeden ihrer Schritte. Verwendet wird dafür die Feldvariable w, die auf nur 100 Elemente dimensioniert ist. Es kann aber sein, daß die Ratte bei ihren ersten Versuchen viel mehr Schritte zurücklegt. Deswegen werden in **Zeile 290** zwei Variablen als Zähler verwendet. Die Variable s zählt die bislang gelaufenen Schritte und wird als Index der Feldvariablen w verwendet, um die Abfolge der Schritte zu erfassen. Da w(s) durch die Di-

Programmbeschreibung
LABYRATT.LRN

mensionierung keinen höheren Index als 99 zuläßt, wird s auf 0 zurückgesetzt, wenn 99 überschritten wird. Die Variable sz zählt ebenfalls die Schritte, aber auch über 99 hinaus.
In **Zeile 300** wird die alte Bewegungsrichtung der Ratte r mit der neuen rn überschrieben.
Zeile 310: Die Variable ww wird erst auf den Wert 1 gesetzt, wenn die Ratte zum ersten Mal den Ausgang gefunden hat. Der Weg wird also erst beim zweiten Versuch mitgeschrieben, weil es praktisch ausgeschlossen ist, daß die Ratte auf Anhieb den kürzesten Weg findet. Sonst wird mit Hilfe des Schrittzählers s als Index die gerade gelaufene Richtung rn in w(s) notiert. Nach dem 99sten Schritt wird die Feldvariable wieder bei w(0) beginnend beschrieben. Damit ist natürlich das Protokoll des Weges nicht mehr vollständig und deshalb unbrauchbar. Bei der Größe der Labyrinthfläche kann aber ausgeschlossen werden, daß der kürzeste Weg hindurch länger als 99 Schritte ist. Läuft die Ratte also 100 oder mehr Schritte, hat sie den kürzesten Weg noch nicht gefunden, und das Wegprotokoll wird gar nicht benötigt.
Nach all diesen mühevollen Überlegungen muß die Ratte erst einmal wieder schnuppern. Dazu springt sie ins Unterprogramm ab **Zeile 16010**, wo die Situation um das Feld herum untersucht wird, auf dem sich die Ratte gerade befindet. Dabei schaut das kluge Tier zuerst in die vier Richtungen: Sind dort Wände (2) oder Wege (0)? Die Ratte denkt nach, indem sie die Zellen in ihrem Gedächtnis danach abfragt, ob sie sich hier eine Sackgasse (2) notiert hat. Die Summe dieser Denkanstrengungen wird an die Variable g übergeben.
Hat g den Wert 6, heißt das, drei der vier Richtungen sind durch Wände oder Sackgassenmarken versperrt. Da die Ratte notwendigerweise aus der vierten Richtung gekommen ist, befindet sie sich offensichtlich am toten Ende einer Sackgasse. Diese Erkenntnis wird in der Flag-Variablen f notiert, dann wird die Bewegungsrichtung umgedreht.
Wenn g kleiner als 4 ist, sind drei Bewegungsrichtungen frei. Die Ratte steht also an einer Abzweigung. Diese Erkenntnis wird in **Zeile 16030** im Flag e vermerkt. Danach erfolgt der Rücksprung ins Hauptprogramm, dessen Bearbeitung bei **Zeile 330** fortgesetzt wird.
Dort werden die Flags e und f abgefragt. Nur wenn beide gesetzt sind, lernt die Ratte. Sonst wird in **Zeile 340** der Merker e (»Ich stehe an einer Abzweigung«) wieder gelöscht und mit **Zeile 350** der nächste Bewegungsschritt eingeleitet.

Programmbeschreibung
LABYRATT.LRN

Wenn die Ratte in eine Sackgasse gelaufen ist, setzt sie an ihrem toten Ende f auf 1 und merkt sich damit: »Ich befinde mich in einer Sackgasse.« Dann dreht sie ihre Bewegungsrichtung um und geht den Weg zurück. Solange sie um sich herum zwei Wände findet ($g = 4$), ist sie noch nicht am Ende der Sackgasse und kann nur dem gegebenen Weg folgen. Steht sie aber an einem Wegknoten ($g < 4 \Rightarrow e = 1$), ist sie soeben aus der Sackgasse herausgetreten. Das gerade überschrittene Feld ist der Eingang zur Sackgasse, und genau dieses Feld wird im Rattengedächtnis rm mit dem Wert 2 markiert.
Das Labyrinth hat eine Baumstruktur, bei der die Wegkreuzungen die Entscheidungsknotenpunkte bilden. Die Ratte bewegt sich zufällig in dieser Baumstruktur. Endet ihr Weg in einer Sackgasse, geht sie den Ast zurück, bis sie an einen Knoten kommt, wo sie ihn anschneidet. Dabei wird aus einem Wegknoten, an dem die Ratte die Wahl zwischen zwei Wegen hatte, ein einfacher Gang, der künftig keine Richtungswahl mehr fordert. Der Weg in die Sackgasse hinein ist nun verschlossen. Auf diese Weise wird der weit verzweigte Baum mehr und mehr zurückgestutzt, bis irgendwann alle toten Äste abgeschnitten sind und nur noch der eine Weg zurückbleibt, der durch das Labyrinth führt.
Die Ratte hat gute Chancen, den Irrgarten nicht vollständig bis in die letzte Sackgasse hinein ablaufen zu müssen. Wenn sie durch algorithmische Orientierung und heuristische Richtungswahl den kürzesten Weg gefunden hat, kann sie die Suche einstellen, auch wenn Bereiche des Baumes noch unerforscht sind.
Und woran erkennt die Ratte, daß sie den kürzesten Weg gefunden hat? Unsere Expertenratte kennt die Regel: Wenn ich in eine Sackgasse gelaufen bin, habe ich einen Umweg gemacht. Den kürzesten Weg hat sie also genau dann gefunden, wenn sie auch nicht ein einziges Mal vor einer Wand gestanden hat. Deswegen merkt sie sich in der Variablen p, wie oft sie in einer Sackgasse kehrtmarsch machen mußte.
Wenn sie nun den Ausgang erreicht hat, und p ist noch immer 0, ist sie den kürzesten Weg gelaufen. In der Feldvariablen w sind alle Schritte dieses Weges gespeichert. Im Unterprogramm »Ausgang erreicht« ab **Programmzeile 20000** wird dann gleich nach **Zeile 20230** verzweigt, wo eine entsprechende Meldung und die Anzahl der benötigten Schritte ausgegeben wird. Drückt der Anwender dann eine Taste, um die Ratte noch einmal durch das Labyrinth zu jagen, führen sie die **Zeilen 20400 bis 20550** mit Hilfe von w(i) Schritt für Schritt den gefundenen Weg zum Ausgang. Wurde der

kürzeste Weg noch nicht gefunden (p>0), wird angezeigt, wie viele Schritte zurückgelegt wurden. Auf Tastendruck startet die Ratte einen neuen Versuch.
Wenn der menschliche Anwender der Ratte auf ihren Irrwegen zuschaut, mag er manches Mal über ihre mühsame Suche lächeln. Der Irrgarten ist relativ klein und überschaubar. Aus der Vogelperspektive am Bildschirm überblickt man leicht, welche Richtung zum Ausgang führt, in welchen Bereichen die Suche sinnlos ist. Aber selbst ein kluger Mensch, der sich innerhalb dieses Labyrinthgemäuers bewegen müßte, könnte nur wahllos in Gänge hineingehen, bis er sie als Sackgasse erkennt, da ihm der Gesamtüberblick fehlt. Wer jemals durch einen Irrgarten gelaufen ist, kennt diesen Unterschied.

Eine prinzipiell noch mögliche Verbesserung: Der Ariadne-Faden

Die einzige Unart, die man der Ratte noch austreiben könnte, besteht darin, daß sie manches Mal einen Weg wählt, der zum Eingang zurückführt, weil sie sich den abgeschrittenen Weg nicht merkt. Das würde aber einem Labyrinthgänger genauso gehen. — Das Programm könnte also noch verbessert werden, wenn ein Ariadnefaden den bereits erkundeten Weg markieren würde.
Schon in dieser Form verwirklicht das Programm Ansätze der KI. Ein Algorithmus im klassischen Sinn durchforstet die gesamte Baumstruktur des Labyrinths und wirft am Ende den Lösungsweg aus. Durch die eingebrachte Heuristik ist die Wahrscheinlichkeit groß, daß nur ein Teil des Baumes abgesucht werden muß, bis der kürzeste Weg gefunden ist.

Spielend gelernt

Fast alle Probleme lassen sich dadurch lösen, daß man einen Weg vom Anfangszustand zu einem definierten Endzustand sucht. Der gesuchte Lösungsweg ergibt sich aus einer Abfolge von Entscheidungen zwischen alternativen Möglichkeiten.
Stellen Sie sich vor, Sie stehen mit Ihrem Wagen auf dem Parkplatz einer fremden Stadt und wollen zum Bahnhof fahren. Das Straßennetz der Stadt ist Ihnen gänzlich unbekannt. Sie haben auch keinen Stadtplan oder ähnliche Hilfsmittel zur Orientierung. Ihre Möglichkeiten beschränken sich darauf, an jeder Kreuzung auf Ihrem Weg die richtige Abzweigung zu wählen. Das Unterfangen scheint sinnlos.

Schon wenn Sie den Parkplatz verlassen, müssen Sie sich entscheiden, ob Sie nach links oder rechts fahren wollen. Die Wahrscheinlichkeit beträgt 50%, daß Sie einen Weg einschlagen, der in die Irre führt. Selbst wenn Sie zufällig die richtige Richtung gewählt haben, stehen Sie an der nächsten Kreuzung wieder vor dem gleichen Problem. Wie viele solcher Kreuzungen Sie vor sich haben, bis das Ziel erreicht ist, wissen Sie auch nicht.
Das Problem ist theoretisch lösbar, indem jede mögliche Strecke ausprobiert wird. Eine davon muß ja schließlich zum Bahnhof führen. Wenn Sie allerdings nur zwanzig Kreuzungen zu bewältigen haben, zwischen denen im Schnitt 100 m liegen, und bei jeder unter drei Richtungen wählen müssen, haben Sie im ungünstigsten Fall etwa 800 km vor sich. — Und diese Aufgabenstellung kann noch als besonders simpel bezeichnet werden.
Da Straßen vernetzt sind und ein Fahren im Kreis möglich ist, taugt ein Baum weit besser als Anschauungsbeispiel: Unzweifelhaft besteht eine direkte Verbindung vom Stamm zu jedem einzelnen Blatt. Bei jeder Gabelung gibt es nur die Alternative, den einen oder den anderen Ast weiter zu verfolgen. Beträgt der mittlere Astweg von Gabelung zu Gabelung 10 cm und trägt der Baum etwa 30 000 Blätter (die Summe der Gabelpunkte ist vernachlässigbar größer), dann muß eine Strecke von rund 90 000 km durchmessen werden, um von der Wurzel aus mit Sicherheit ein bestimmtes Blatt zu finden. Bei diesem Baumproblem sind $30\,000^2$ Entscheidungen zu treffen, bis die Zielvorgabe endgültig erreicht ist, das sind weniger als 10^9. Schon ein Modell wie das Schachspiel überragt es um ein Vielfaches. Man schätzt die Anzahl seiner möglichen Stellungen auf 10^{120}.

Schon das »Bahnhofssuche«-Problem sprengt ohne strukturierende Maßnahmen den Rahmen vernünftiger Rechenzeiten

Selbst die schnellste Maschine könnte nicht in einem vernünftigen Zeitrahmen diese Vielfalt von Möglichkeiten durcharbeiten. Kein Mensch würde so an diese Aufgabe herangehen. Ein guter Schachspieler bringt das Problem, welchen Zug er als nächsten machen soll, auf ein überschaubares Maß, weil sein Erfahrungswissen intuitiv einfließt.
Um noch einmal auf das Parkplatzproblem zurückzukommen. Wenn der Fahrer weiß, daß der Bahnhof in südlicher Richtung liegt, kann er sich an der Sonne orientieren. Schon beim Verlassen des Parkplatzes kann er durch Einschlagen der entsprechenden Richtung den Suchaufwand halbieren. Bei seinen weiteren Richtungsentscheidungen wird er sich nach Süden orientieren und von Zeit zu Zeit überprüfen, ob die eingeschlagene Richtung mit der Zielrichtung überein-

stimmt, um entstandene Abweichungen bei weiteren Entscheidungen — soweit möglich — zu korrigieren.
Das systematisch sture Absuchen (algorithmisch) wird ersetzt durch schrittweises Versuchen und Überprüfen, für das formulierte Regeln bereitgehalten werden (heuristisch). Bei Spielbäumen besteht eine der heuristischen Methoden darin, die erreichten Spielsituationen zu bewerten. Beim Durchsuchen des Entscheidungsbaumes wird nach jeder Verzweigung die damit erreichte Bewertungsziffer ermittelt. Die jeweils beste Zwischenwertung wird zum Maßstab, nach dem alle Gabelungen verworfen werden können, die zu einer schlechteren Bewertungsziffer führen. Nachdem erst einmal irgendeine Lösung gefunden ist, sucht das Programm nur noch nach einer besseren. Sowie ein Lösungsweg unter den Richtwert absackt, wird seine Bearbeitung verworfen.

Warum die Großmeister den Schachprogrammen überlegen sind

Strategien, die ein Großmeister vielleicht anwenden mag, bei denen vorübergehend eine nach den angewendeten Maßstäben schlechter zu bewertende Spielsituation entsteht, aus der sich aber nach mehreren Folgezügen entsprechend größere Vorteile ergeben, können damit natürlich nicht gefunden werden. Bislang ist ein Großmeister ja auch noch jedem Schachprogramm überlegen.
BASIC würde sicherlich den Mund etwas voll nehmen, versuchte es sich an Schach. Im folgenden Programm geht es deshalb um ein einfacheres Spiel.
Das Brett besteht aus nur 3 x 3 Feldern. In der obersten Reihe stehen die drei dunklen Figuren, die vom Computer geführt werden. Die mittlere Feldreihe ist leer. In der unteren Reihe stehen die drei hellen Figuren des Spielers.
Die Figuren bewegen sich bei jedem Zug nur um ein Feld und immer nur geradeaus oder schräg vorwärts, nie seitwärts oder rückwärts. Auf ein freies Feld kann nur geradeaus gezogen werden, auf ein vom Gegner besetztes Feld nur diagonal. Der gegnerische Stein wird dabei aus dem Spiel geschlagen. Felder, die von eigenen Steinen besetzt sind, können nicht betreten werden. Steht eine Figur auf der gegnerischen Grundlinie, ist sie bewegungsunfähig. Die Figuren bewegen sich also wie die Bauern des Schachspiels. Eine Einschränkung besteht darin, daß sie auf der Grundlinie nicht gewandelt werden. Der Benutzer eröffnet das Spiel. Die Partie verliert, wer an der Reihe ist und keinen zulässigen Zug mehr machen kann.
Es ist natürlich denkbar, alle möglichen Spielsituationen zu erfassen und den jeweils besten Zug für den

Computer im Programm festzuschreiben. Obwohl das Spiel sehr einfach angelegt ist, gibt es aber reichlich viele Kombinationen.
MOUSCHEX.LRN geht deshalb einen anderen Weg. Im Programm sind nur die Regeln für zulässige Züge verankert (**Zeilen 340 bis 360**). Der Computer erfaßt die Figurenkonstellation auf dem Brett, ermittelt dann alle für ihn möglichen Züge und wählt einen davon zufällig aus. Wir ersparen es dem Rechner, alle möglichen Züge so weit zu verfolgen, bis sie zu einem Ende der Partie führen und sich in Sieg oder Niederlage als richtig oder falsch erweisen. Da wir nur zufällig wählen lassen und nicht einmal Erfahrungswerte eingeben, mit welcher Wahrscheinlichkeit ein bestimmter Zug zum Sieg führt, macht der Rechner anfangs viele dumme Züge. Anfangs.

Die einfachste Lernmethode: Vermeide, was erfahrungsgemäß zu Mißerfolgen führt

Das Expertenwissen kommt erst später ins Spiel. Nach fachlicher Analyse können wir feststellen, daß der letzte Zug des Rechners, der zu seiner Niederlage geführt hat, ein schlechter Zug sein muß. Wir lassen den Computer intern eine Bibliothek aller jemals erfahrenen Figurenkonstellationen und der von ihm daraufhin (zufällig) gewählten Züge anlegen. Führt ein Zug zur Niederlage, wird er in der Bibliothek entsprechend markiert und später nie mehr ausgeführt.
Einzelne Spiele können als ein bestimmter Pfad durch das Gewirr des Entscheidungsbaumes betrachtet werden, der alle möglichen Spiele enthält. Durch die gewählte Methode werden die Äste dieses Baumes von ihren äußersten Enden, also den Figurenkonstellationen der beendeten Partien, rückwärts verfolgt. An allen Knotenpunkten wird die Verzweigung abgetrennt, die zur Niederlage geführt hat.
In der Praxis sieht das so aus: Eine Partie wurde gespielt, der Computer hat verloren; er streicht seinen letzten Zug als Lösungs- bzw. Gewinnmöglichkeit. Kommt es im folgenden Spiel zu einer gleichen Figurenkonstellation, wählt er einen anderen Zug aus. Er verfolgt also einen anderen Ast des Entscheidungsbaumes. Stellt er in dieser Situation jedoch fest, daß ihm kein gewinnträchtiger Zug mehr bleibt, daß er also verloren hat, so löscht er wiederum den zuvor gemachten Zug. Es wird bei folgenden Partien zu dieser Situation nicht mehr kommen, da bereits am vorgeordneten Entscheidungsknoten ein entsprechender Wegweiser aufgestellt wurde.
Während der menschliche Spieler anfangs fast jede Partie gewinnt, hat er mit der Zeit immer mehr Schwierigkeiten, sich gegen den Rechner durchzusetzen.

Jede Partie besteht nur aus wenigen Zügen. So lassen sich schnell einige Dutzend Spiele ausfechten. Um die Auswirkungen der Intelligenz des Programmes schneller zu spüren, können Sie jedesmal mit dem gleichen Zug eröffnen. Den ersten sanften KI-Schock erlebt man, wenn der Rechner meldet, daß er verloren hat, obwohl er noch einen zulässigen Zug machen könnte. Er »weiß« eben, daß dieser eine Zug nichts mehr am Spielausgang ändern kann.

Kann man gar nicht mehr gegen den Rechner gewinnen, ist es ratsam, die Eröffnung zu wechseln. Dadurch wird ein anderer Bereich des Entscheidungsbaumes betreten, in dem das Programm noch nicht gelernt hat. Es ist problemlos möglich, die vom Rechner gesammelten Erfahrungen dauerhaft auf einem Datenträger zu speichern und so mit der Zeit ein gewinnsicheres System aufzubauen. Dieses Programm verzichtet darauf, weil sein Reiz gerade darin liegt, den Benutzer miterleben zu lassen, wie der Computer von Partie zu Partie besser spielt. Spätestens nach 100 Partien wird er praktisch keine Fehler mehr machen. Und wem macht es schon Spaß zu spielen, wenn er vorweg weiß, daß er verlieren wird?

Ein bewußter Verstoß gegen die KI-Regeln: Dieses Programm »vergißt« beim Ausschalten des Rechners seine Erfahrungen

Programm MOUSCHEX.LRN

```
 1 REM MOUSCHEX.LRN
 10 RANDOMIZE TIME
 20 MODE 1:CLS:GOSUB 60010
 30 GOSUB 10010
 90 GOSUB 10500
 100 INPUT;"Bitte geben Sie Ihren Zug ein
 :                alte Position der Figur <RETU
 RN>   ",a:PRINT
 110 q=a:GOSUB 13010
 120 ax=qx:ay=qy
 130 IF b(ax,ay)<>-1 THEN CLS:PRINT"     D
 ORT STEHT KEINER IHRER STEINE !
      Bitte neu eingeben":FOR j=0 TO 100
 0:NEXT j:GOTO 100
 140 INPUT;"neue Position der Figur <RETU
 RN>   ",z
 150 q=z:GOSUB 13010
 160 zx=qx:zy=qy
 170 IF ax=zx AND b(zx,zy)<>0 THEN CLS:PR
 INT "             UNZULASSIGER ZUG!
                Bitte neu eingeben":FOR j
 =0 TO 1000:NEXT j:GOTO 100
 180 IF ax<>zx AND b(zx,zy)<>1 THEN CLS:P
 RINT "             UNZULASSIGER ZUG!
                Bitte neu eingeben":FOR
 j=0 TO 1000:NEXT j:GOTO 100
 190 IF ABS(ax-zx)>1 OR ABS(ay-zy)>1 THEN
  CLS:PRINT "             UNZULASSIGER ZUG!
                      Bitte neu eingeben
```

Programm
MOUSCHEX.LRN

```
":FOR j=0 TO 1000:NEXT j:GOTO 100
200 b(zx,zy)=-1:v=zx*3+1:w=zy*3+1:GOSUB
11010
210 b(ax,ay)=0:v=ax*3+1:w=ay*3+1:GOSUB 1
2010
220 GOSUB 14010
300 REM > Computer-Zug <
310 l=0
320 FOR j=1 TO 3:FOR i=1 TO 3
330 f=i+(j-1)*3
340 IF b(i,j)=1 AND b(i,j+1)=0 THEN mz(l
,0)=f:mz(l,1)=1:l=l+1
350 IF b(i,j)=1 AND b(i+1,j+1)=-1 THEN m
z(l,0)=f:mz(l,1)=2:l=l+1
360 IF b(i,j)=1 AND b(i-1,j+1)=-1 THEN m
z(l,0)=f:mz(l,1)=3:l=l+1
370 NEXT i:NEXT j
380 IF l=0 GOTO 20010
400 FOR k=0 TO l-1
410 f=mz(k,0):r=mz(k,1)
420 GOSUB 15010
430 FOR j=0 TO n
440 IF zf(j,0)=ac AND zf(j,1)=as AND zf(
j,2)=zc AND zf(j,3)=1 THEN p=p+1
450 IF zf(j,0)=ac AND zf(j,1)=as AND zf(
j,2)=zc THEN u=1
460 NEXT j
500 IF u=1 GOTO 520
510 zf(n,0)=ac:zf(n,1)=as:zf(n,2)=zc:n=n
+1
520 u=0
530 NEXT k
600 IF l=p GOTO 20010
610 p=0
700 d=INT(RND(1)*l)
710 f=mz(d,0):r=mz(d,1)
720 GOSUB 15010
730 FOR j=0 TO n
740 IF zf(j,0)=ac AND zf(j,1)=as AND zf(
j,2)=zc AND zf(j,3)=1 THEN u=1
750 NEXT j
760 IF u=1 THEN u=0:GOTO 700
770 q=a
780 GOSUB 13010
790 ax=qx:ay=qy
800 q=z
810 GOSUB 13010
820 zx=qx:zy=qy
830 PEN #1,2
840 b(zx,zy)=1:v=zx*3+1:w=zy*3+1
850 GOSUB 11010
860 b(ax,ay)=0:v=ax*3+1:w=ay*3+1
870 GOSUB 12010
880 PEN #1,1
890 vc=ac:ac=zc
900 GOSUB 16010
910 IF c=0 GOTO 21010
920 CLS:GOTO 100
10000 REM > Bildschirmgestaltung <
```

Programm
MOUSCHEX.LRN

```
10010 DEFINT a-z
10020 DIM zf(100,3),zk(100,2):m=-1
10030 BORDER 0:INK 0,0:INK 1,25:INK 2,4:
INK 3,10:CLS
10040 WINDOW #1,14,28,3,17
10050 WINDOW #2,2,5,5,7:PAPER #2,1:CLS #
2
10060 WINDOW #3,37,40,5,7:PAPER #3,2:CLS
 #3
10070 PEN 3
10080 LOCATE 18,4:PRINT CHR$(198):LOCATE
 21,4:PRINT CHR$(198):LOCATE 24,4:PRINT
CHR$(198)
10090 LOCATE 18,3:PRINT"4":LOCATE 21,3:P
RINT"5":LOCATE 24,3:PRINT"6"
10100 LOCATE 18,16:PRINT CHR$(196):LOCAT
E 21,16:PRINT CHR$(196):LOCATE 24,16:PRI
NT CHR$(196)
10110 LOCATE 18,17:PRINT"4":LOCATE 21,17
:PRINT"5":LOCATE 24,17:PRINT"6"
10120 LOCATE 14,7:PRINT"1":LOCATE 14,10:
PRINT"2":LOCATE 14,13:PRINT"3"
10130 LOCATE 15,7:PRINT CHR$(197):LOCATE
 15,10:PRINT CHR$(197):LOCATE 15,13:PRIN
T CHR$(197)
10140 LOCATE 27,7:PRINT CHR$(199):LOCATE
 27,10:PRINT CHR$(199):LOCATE 27,13:PRIN
T CHR$(199)
10150 LOCATE 28,7:PRINT"1":LOCATE 28,10:
PRINT"2":LOCATE 28,13:PRINT"3"
10160 LOCATE 2,3:PRINT"Punktestand":LOCA
TE 30,3:PRINT"Punktestand"
10170 LOCATE 2,4:PEN 1:PRINT"Spieler"
10180 LOCATE 33,4:PEN 2:PRINT"Computer"
10190 PEN #2,3:LOCATE #2,1,2:PRINT #2,ps
:PEN #3,3:LOCATE #3,1,2:PRINT #3,pc
10200 RETURN
10500 ac=7
10510 PEN #1,2
10520 v=4:w=4:GOSUB 11010
10530 v=7:w=4:GOSUB 11010
10540 v=10:w=4:GOSUB 11010
10550 PEN #1,1
10560 v=4:w=10:GOSUB 11010
10570 v=7:w=10:GOSUB 11010
10580 v=10:w=10:GOSUB 11010
10590 v=4:w=7:GOSUB 12010
10600 v=7:w=7:GOSUB 12010
10610 v=10:w=7:GOSUB 12010
10620 FOR x=0 TO 4:FOR y=0 TO 4:b(x,y)=3
:NEXT y:NEXT x
10630 FOR x=1 TO 3:b(x,1)=1:NEXT x
10640 FOR x=1 TO 3:b(x,2)=0:NEXT x
10650 FOR x=1 TO 3:b(x,3)=-1:NEXT x
10660 LOCATE 21,10:PEN 3:PRINT CHR$(200)
10670 WINDOW #0,1,40,19,25
10680 RETURN
11000 REM > Figur ins Brett setzen <
11010 FOR i=0 TO 2:FOR j=0 TO 2
```

Programm
MOUSCHEX.LRN

```
11020 t=i+3*j
11030 LOCATE #1,v+i,w+j:PRINT #1,CHR$(16
0+t)
11040 NEXT j:NEXT i
11050 RETURN
12000 REM > Figur im Brett löschen <
12010 FOR i=0 TO 2:FOR j=0 TO 2
12020 LOCATE #1,v+i,w+j:PRINT #1," "
12030 NEXT j:NEXT i
12040 RETURN
13000 REM > Input-Index-Zuordnung <
13010 IF q=14 OR q=41 THEN qx=1:qy=1
13020 IF q=15 OR q=51 THEN qx=2:qy=1
13030 IF q=16 OR q=61 THEN qx=3:qy=1
13040 IF q=24 OR q=42 THEN qx=1:qy=2
13050 IF q=25 OR q=52 THEN qx=2:qy=2
13060 IF q=26 OR q=62 THEN qx=3:qy=2
13070 IF q=34 OR q=43 THEN qx=1:qy=3
13080 IF q=35 OR q=53 THEN qx=2:qy=3
13090 IF q=36 OR q=63 THEN qx=3:qy=3
13100 RETURN
14000 REM > Stellungskennzahl Spieler <
14010 vs=as:as=0
14020 FOR j=1 TO 3:FOR i=1 TO 3
14030 READ e
14040 IF b(i,j)=-1 THEN as=as+e
14050 NEXT i:NEXT j
14060 RESTORE:RETURN
15000 REM > Feld/Zug-Umwandlung <
15010 IF f=1 THEN a=14:ax=1:ay=1
15020 IF f=2 THEN a=15:ax=2:ay=1
15030 IF f=3 THEN a=16:ax=3:ay=1
15040 IF f=4 THEN a=24:ax=1:ay=2
15050 IF f=5 THEN a=25:ax=2:ay=2
15060 IF f=6 THEN a=26:ax=3:ay=2
15070 IF f=7 THEN a=34:ax=1:ay=3
15080 IF f=8 THEN a=35:ax=2:ay=3
15090 IF f=9 THEN a=36:ax=3:ay=3
15100 IF r=1 THEN zx=ax:zy=ay+1:z=a+10
15110 IF r=2 THEN zx=ax+1:zy=ay+1:z=a+11
15120 IF r=3 THEN zx=ax-1:zy=ay+1:z=a+9
15200 FOR j=1 TO 3:FOR i=1 TO 3:bp(i,j)=
b(i,j):NEXT i:NEXT j
15210 bp(ax,ay)=0:bp(zx,zy)=1
15220 zc=0
15300 FOR j=1 TO 3:FOR i=1 TO 3
15310 READ e
15320 IF bp(i,j)=1 THEN zc=zc+e
15330 NEXT i:NEXT j
15340 RESTORE:RETURN
16000 REM > Summe Spielerzüge <
16010 c=0
16020 FOR j=1 TO 3:FOR i=1 TO 3
16030 IF b(i,j)=-1 AND b(i,j-1)=0 THEN c
=c+1
16040 IF b(i,j)=-1 AND b(i+1,j-1)=1 THEN
 c=c+1
16050 IF b(i,j)=-1 AND b(i-1,j-1)=1 THEN
 c=c+1
16060 NEXT i:NEXT j
```

Programm
MOUSCHEX.LRN

```
16070 RETURN
20000 > Computer verliert <
20010 FOR j=0 TO n
20020 IF zf(j,0)=vc AND zf(j,1)=vs AND z
f(j,2)=ac THEN zf(j,3)=1
20030 NEXT j
20040 CLS:PEN 2
20050 PRINT"                 Ich habe verloren
!"
20060 PRINT:PRINT" Aber ich habe dazugel
ernt. Die nächste            Partie spiele
ich besser.            Drücken Sie <T> und
 ich beweise es.";
20070 ps=ps+1:LOCATE #2,1,2:PRINT #2,ps
20080 IF INKEY$="" GOTO 20080
20090 CLS:PEN 3:GOTO 90
21000 REM > Spieler verliert <
21010 CLS:PEN 1
21020 PRINT"       Tut mir leid, Sie haben
verloren!"
21030 PRINT:PRINT"    Ich habe dazugelern
t. Die nächste           Partie spiele ich
 noch besser.        Drücken Sie <T> und
ich beweise es.";
21040 pc=pc+1:LOCATE #3,1,2:PRINT #3,pc
21050 IF INKEY$="" GOTO 20080
21060 CLS:PEN 3:GOTO 90
21070 DATA 1,2,4,8,16,32,64,128,256
60000 REM > Umlaute, Sonderzeichen <
60010 SYMBOL AFTER 91
60020 SYMBOL 91,66,24,60,102,126,102,102
,0
60030 SYMBOL 92,130,56,108,198,198,108,5
6,0
60040 SYMBOL 93,66,0,102,102,102,102,60,
0
60050 SYMBOL 123,68,0,120,12,124,204,118
,0
60060 SYMBOL 124,36,0,60,102,102,102,60,
0
60070 SYMBOL 125,36,0,102,102,102,102,62
,0
60080 SYMBOL 126,60,102,124,102,102,102,
108,96
60090 SYMBOL 160,0,0,1,7,14,29,27,55
60100 SYMBOL 161,0,126,255,191,255,255,2
55,255
60110 SYMBOL 162,0,0,128,224,240,248,120
,188
60120 SYMBOL 163,55,111,111,111,127,127,
127,63
60130 SYMBOL 164,223,187,189,125,253,251
,231,255
60140 SYMBOL 165,220,254,238,238,238,238
,254,220
60150 SYMBOL 166,63,31,31,15,7,3,0,0
60160 SYMBOL 167,255,255,254,255,255,255
,126,0
60170 SYMBOL 168,188,120,248,240,224,128
,0,0
```

Programm
MOUSCHEX.LRN

```
60180 SYMBOL 200,196,204,216,243,243,216
,204,196
60190 RETURN
65000 REM > Inhalt des Computer-Gedächtn
isses <
65010 REM Wenn Sie das Programm mit <ESC
> unterbechen und dann GOTO 65000 eingeb
en, werden alle gelernten Züge auf dem D
rucker ausgegeben.
65020 FOR j=0 TO n
65030 PRINT #8,j;zf(j,0);zf(j,1);zf(j,2)
;zf(j,3)
65040 NEXT j
```

Programmbeschreibung
MOUSCHEX.LRN

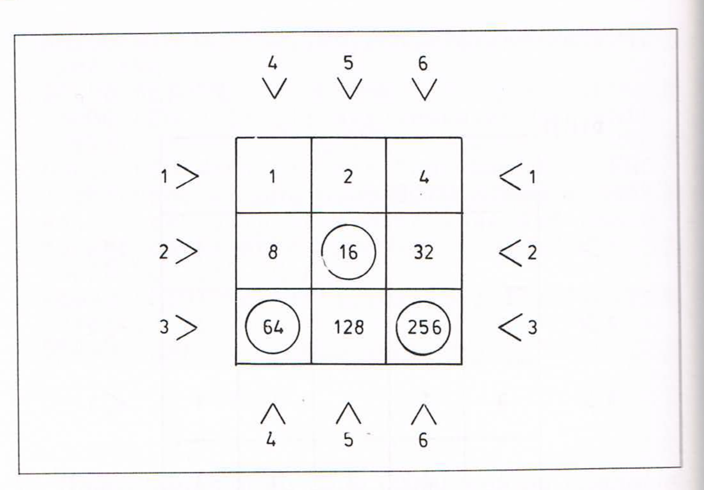
Im **Unterprogramm ab Zeile 60000** werden Elemente des Zeichensatzes verändert, um deutsche Sonderzeichen und die Spielfiguren darzustellen, die sich aus 3 x 3 Zeichen zusammensetzen.
Das **Unterprogramm ab Zeile 10000** gestaltet den Bildschirm. Das Spielfeld in der Mitte ist 3 x 3 Felder groß, von denen jedes 3 x 3 Schreibstellen bedeckt. Am Rande des Brettes befinden sich Koordinaten, um die Spielfelder zu bezeichnen. Die Zeilen sind mit 1 bis 3, die Spalten mit 4 bis 6 numeriert.
Der Spieler zieht, indem er die Koordination des Feldes eingibt, auf dem die Figur steht, die er bewegen will. Reihe und Spalte können in beliebiger Folge eingegeben werden, also zum Beispiel 35 oder 53 für das mittlere Feld der untersten Reihe. Danach wird das Zielfeld auf gleiche Weise benannt. Die Eingaben des Benutzers werden in den **Zeilen 100 bis 190** aufgenommen und mit den Regeln für zulässige Züge verglichen.
Nun setzt das Programm den Spielerzug um. Dabei werden zwei Operationen durchgeführt. In der Feldvariablen b(n,m) notiert das Programm den Zustand jedes einzelnen Spielfeldes. Das Array ist 5 x 5 Elemente groß, da auch Informationen darüber erfaßt werden müssen, wo der Rand des Brettes erreicht ist. Ein Element des Arrays ist 0, wenn das Spielfeld leer ist, 1, wenn eine Figur des Computers darauf steht, 2, wenn eine Figur des Spielers darauf steht, und 3, wenn es außerhalb des Spielfeldes (= Rand) liegt. Die Abbildung zeigt die Grundaufstellung. Nach jedem Zug muß der Inhalt des Arrays b(i,j) entsprechend verändert werden. Dies geschieht in den **Zeilen 200 und 210**.
Außerdem werden die Daten ermittelt, die notwendig sind, um die Grafik auf dem Bildschirm entsprechend zu verändern. Dies geschieht in den **Unterprogrammen ab den Zeilen 11000 und 12000**.

b(i,j)

		4 ∨	5 ∨	6 ∨		
	3	3	3	3	3	
1 >	3	2	2	2	3	< 1
2 >	3	0	0	0	3	< 2
3 >	3	1	1	1	3	< 3
	3	3	3	3	3	
		∧ 4	∧ 5	∧ 6		

Spielfeld und Steine werden in Zahlenwerte umgesetzt und den Elementen der Feldvariablen b(i,j) zugeordnet. Felder jenseits des Spielfeldes haben den Wert 3. Sie markieren den Rand. Leere Felder werden durch eine 0, Felder mit einem Spielerstein durch 1 und Felder mit einem Computerstein durch 2 dargestellt. Über die Zahlen bei den kleinen Pfeilen kann der Benutzer jedes der neun Felder des Spielplans benennen.

Abschließend wird die Stellung der Spielerfiguren auf dem Brett zu einer Kennzahl zusammengefaßt. Zu diesem Zweck sind die neun Spielfelder mit Zahlenwerten belegt, die den Zweierpotenzen entsprechen, also 1, 2, 4 usw. Die Abbildung auf S. 100 zeigt die gewählte Verteilung.
Die Stellungskennzahl ist eine 9-Bit-Information: Neun Felder können entweder besetzt oder nicht besetzt sein. Steht eine Spielerfigur auf einem Feld, wird das entsprechende Bit gesetzt. In der Grundstellung belegen die drei Figuren des Spielers die untere Zeile. Die Bits 6, 7 und 8 sind gesetzt, die Stellungskennzahl ist dezimal $64+128+256=448$. Würde der Spieler die mittlere Figur ziehen, wären danach die Bits 4, 6 und 8 gesetzt. Die Stellungskennzahl wäre dann $16+64+256=336$. Die Berechnung der Kennzahl erfolgt im **Unterprogramm ab Zeile 14000**. Die Stellungskennzahl des Spielers wird in der Variablen as aufgehoben.
Danach ist der Rechner an der Reihe, seinen Zug zu machen. Zuerst stellt er fest, wie viele Züge er in der vorliegenden Situation überhaupt machen kann. Die Menge der möglichen Züge wird von der Variablen l gezählt,



Die Position aller ein bis drei Steine einer Partei wird durch einen einzigen Zahlenwert beschrieben. Dazu wird jedem Spielfeld eine Zweierpotenz zugeordnet. Dann werden die Werte der Felder, die von den Steinen einer Partei besetzt sind, addiert.

die möglichen Züge selbst im Array mz(n,m) notiert. Dabei numeriert n die möglichen Züge fortlaufend. In m wird der Zug selbst erfaßt; m = 0 notiert die Feldnummer. Die Felder sind zu diesem Zweck von links nach rechts und Zeile um Zeile von 1 bis 9 durchnumeriert. Die mögliche Zugrichtung erfaßt m = 1. Sie wird durch 1 beschrieben, wenn vorwärts gezogen, oder durch 2 bzw. 3, wenn seitlich geschlagen werden kann.
Findet das Programm keinen zulässigen Zug, hat es verloren und verzweigt ins **Unterprogramm ab Zeile 20000**. In diesem Unterprogramm ist die FOR-NEXT-Schleife in den **Zeilen 20010 bis 20030** wichtig. Hier geht der Rechner alle ihm bereits bekannten Zugfolgen durch. Zugfolgen, die er im Verlauf des Spieles kennenlernt, werden in der Feldvariablen zf(n,m) gespeichert, wobei wiederum n alle gelernten Zugfolgen durchnumeriert und m die Zugfolge selbst erfaßt; m = 0 enthält die Stellungskennzahl der Computerfiguren vor dem Zug (vc), m = 1 die Kennzahl der Spielerfiguren vor dem Zug (vs); m = 2 enthält die Kennzahl der Computerfiguren nach dem Zug (ac). Damit ist ein Zug für den Rechner eindeutig beschrieben. Wenn er den gesuchten Zug in seinem Gedächtnis gefunden hat, schreibt er in m = 3 eine 1 und merkt sich damit, daß dieser Zug zur Niederlage geführt hat.
Zurück ins Hauptprogramm: Hat der Rechner mögliche Züge gefunden, geht er sie im nächsten Arbeitsschritt alle durch (ab **Zeile 400**). Kennt er einen Zug bereits (**Zeile 440**), überprüft er in zf(n,3), ob es sich um einen Ver-

Programmbeschreibung
MOUSCHEX.LRN

lierzug handelt. Alle gefundenen Verlierzüge zählt er in der Variablen p. Wenn er den Zug findet (**Zeile 450**), merkt er sich das mit u = 1, damit er ihn nicht in seine Bibliothek der Züge aufnimmt, was sonst in **Zeile 510** erfolgt.
Ist die Summe der möglichen Züge (l) gleich der Summe der gefundenen Verlustzüge (p), hat der Rechner zweifellos verloren. **Zeile 600** überprüft diesen Sachverhalt.
Nachdem sich das Programm bis hierhin durchgearbeitet hat, kann der Rechner endlich seinen Zug ausführen. Dazu wird aus der Menge der gefundenen möglichen Züge (l) ab **Zeile 700** zufällig (d) einer ausgewählt. Die Variable mz(d,m) enthält für m = 0 die laufende Nummer des Spielfeldes und für m = 1 die Zugrichtung.
Wurde auf diese Weise ein möglicher Zug zufällig ausgewählt, muß in der Bibliothek der bekannten Züge (zf) nachgesehen werden, ob er als Verlierzug eingetragen ist (**Zeile 740**). Ist das der Fall, wird so lange per Zufall ein anderer möglicher Zug gewählt, bis einer gefunden wird, der noch nicht zur Niederlage geführt hat.
Danach müssen die laufende Spielfeldnummer und die Zugrichtung in Werte umgesetzt werden, mit denen die Eintragungen im imaginären Spielbrett b(n,m) und die Bildschirmgrafik geändert werden können. Das erledigt das **Unterprogramm ab Zeile 15000**, in dem auch die Stellungskennzahl für die Computerfiguren errechnet wird.
Nachdem der Computer seinen Zug durchgeführt hat, überprüft er im **Unterprogramm ab Programmzeile 16000**, ob der Spieler noch einen zulässigen Zug machen kann, denn sonst hat der verloren. Anschließend übergibt das Programm die Kontrolle an den Benutzer, damit er seinen nächsten Zug eintippen kann.
Ist eine Partie beendet, kann durch Betätigen einer Taste die nächste Partie eröffnet werden. Auf dem Bildschirm erscheint die Grundaufstellung der Figuren.
Wenn Sie den Programmlauf unterbrechen (zweimal <ESC> drücken) und in Mode 1 zurückschalten, können Sie sich den Inhalt des Rechnergedächtnisses ausgeben lassen. Der Ausdruck zeigt die Züge, die das Programm nach zwanzig Spielen gelernt hatte; es sind 55. Die erste Zahl jeder Zeile ist die laufende Nummer, es folgen Computerstellungskennzahl vor dem Zug, Spielerkennzahl vor dem Zug, Computerkennzahl nach dem Zug und zuletzt eine 1, wenn dieser Zug zum Verlust der Partie führt.

```
0  7  392  21  1
1  7  392  13  0
2  7  392  35  0
3  13  272  28  0
4  13  272  41  0
5  13  272  25  0
6  13  272  69  0
7  28  16  40  0
8  28  16  24  0
9  28  16  68  0
10  13  160  69  0
11  13  160  133  0
12  35  296  17  0
13  35  296  33  0
14  35  296  9  0
15  7  336  14  1
16  7  336  22  0
17  7  336  35  1
18  7  336  19  0
19  14  324  24  1
20  22  80  34  0
21  22  80  18  0
22  34  24  40  0
23  34  24  258  1
24  40  2  96  0
25  40  2  264  0
26  35  321  48  1
27  19  80  10  0
28  19  80  18  1
29  22  272  34  0
30  22  272  18  0
31  19  96  26  0
32  19  96  49  0
33  19  96  131  0
34  19  96  67  0
35  49  16  40  0
36  49  16  48  0
37  49  16  257  0
38  22  96  52  0
39  22  96  134  0
40  22  96  70  0
41  52  16  48  0
42  52  16  260  1
43  22  264  28  0
44  22  264  50  0
45  22  264  134  0
46  22  264  262  0
47  134  257  148  1
48  134  257  162  0
49  26  48  40  0
50  26  48  66  0
51  67  2  72  0
52  70  2  96  0
53  19  272  10  0
54  19  272  18  1
55  0  0  0  0
```

Programmbeschreibung MOUSCHEX.LRN

Zum Zeitpunkt des Ausdrucks lautete das Spielergebnis der einzelnen Partien 10:10. Bei einer Eröffnung mit der mittleren Figur (35 nach 25) war das Programm jetzt kaum noch zu schlagen.
Es würde den Rahmen dieses Buches sprengen, das Programm wie die früheren Beispiele Zeile um Zeile zu dokumentieren. Deshalb folgt eine Liste der wichtigsten verwendeten Variablen:

b(n,m): In dieser 5 x 5 Elemente großen Feldvariablen wird der Zustand der Spielfelder notiert. Damit das Programm die Randfelder erkennen kann, grenzen diese an Elemente, die einen entsprechenden Wert (3) halten. Spielfelder, die leer sind, werden durch den Wert 0 dargestellt. Steht eine Spielfigur des Computers auf einem bestimmten Feld, wird das entsprechende Element der Feldvariablen auf 1 gesetzt, handelt es sich um eine Spielerfigur, auf 2.

zf(n,m): In dieser Feldvariablen notiert das Programm alle neuen Zugfolgen, die es während des Programmlaufes kennenlernt. In der Dimension n werden die Zugfolgen fortlaufend numeriert. Der Wert von n entspricht immer der Anzahl der bereits gelernten Zugfolgen zu einem bestimmten Zeitpunkt. Die Dimension m nimmt die Zugfolge selbst auf. Eine Zugfolge wird bestimmt durch die Figurenkonstellation, die der Rechner vorfindet, und durch diejenige, die sich nach seinem Zug ergibt. Da sich eine mögliche Änderung der Figurenkonstellation des Benutzers nach dem Zug (durch Schlagen) zwangsläufig durch den Zug selbst ergibt, braucht sie nicht erfaßt zu werden. In m = 0 wird die Figurenkennzahl der Computerfiguren vor dem Zug (vz), in m = 1 die Spielerkennzahl vor dem Zug (vz) und in m = 2 die Computerkennzahl nach dem Zug (ac) erfaßt; m = 3 ist vorerst 0. Wird diese Zugfolge später als verlustbringend erkannt, wird hier eine 1 abgelegt.

mz(n,m): Dieses Array nimmt vorübergehend alle in einer bestimmten Situation möglichen Züge auf. Nach den Spielregeln gibt es keine Figurenkonstellation, in der mehr als sechs Züge möglich wären. In der Dimension n werden die möglichen Züge durchnumeriert. In der Dimension m erfaßt m = 0 die Nummer des Spielfeldes, auf der eine Computerfigur steht, und m = 1 die Richtung, in die sie bewegt werden kann. Es sind nur drei Richtungen möglich: vorwärts ziehen (1), seitwärts nach links schlagen (2) und seitwärts nach rechts schlagen (3).

Ergeben sich für eine einzige Figur mehrere Zugmöglichkeiten, so werden sie getrennt unter verschiedenen laufenden Nummern (n) registriert.

l zählt die Menge der jeweils möglichen Züge.

p zählt, wie viele der möglichen Züge »verriegelt« sind, weil sie zum Verlust führen.

r beschreibt die möglichen Zugrichtungen von 1 bis 3.

ax,ay; zx,zy: Nachdem vom Spieler oder vom Computer ein Zug gemacht wurde, müssen die Inhalte der Elemente von b(n,m) verändert werden. Die beiden Variablen ax und ay beschreiben die »Position« der gezogenen Figur in b(n,m) vor dem Zug, zx und zy die danach.

a,z: Wenn der Spieler einen Zug machen soll, muß er die Koordinaten des Feldes eingeben, auf dem der zu bewegende Stein steht. Die Koordinaten werden als eine einzige, zweistellige Zahl übernommen und der Variablen a zugeordnet. Dann muß der Spieler das Feld angeben, auf das die Figur ziehen soll. Diese Koordinatenzahl übernimmt z.

ac,as; zc,zs: Nach jedem Zug muß sich das Programm ein Bild von der Stellung aller Figuren auf dem Brett machen. Das geschieht, indem für jedes Feld notiert wird, ob eine Figur darauf steht oder nicht. Jedem Feld ist ein Zahlenwert (fortlaufende Zweierpotenzen) zugeordnet. Die Zahlenwerte aller Felder, auf denen eine Figur steht, werden zu einer Stellungskennzahl addiert. Stellungskennzahlen werden für die Computerfiguren (ac) und die Spielerfiguren (as) getrennt ermittelt. Die Variablen ac und as enthalten jeweils die aktuellen Kennzahlen, also die nach einem Zug; zc und zs jene der vorhergehenden Stellung.

v,w: Aus der alten und neuen Position einer gezogenen Figur muß die Bildschirmposition (in window # 1) errechnet werden, damit die Veränderung auch grafisch auf dem Bildschirm sichtbar werden kann. Die Variablen v und w stellen die entsprechende Bildschirmposition dar.

pc,ps halten den laufenden Punktestand für Spieler (ps) und Rechner (pc).

Schlaumeier

Ich sehe was, was du nicht siehst. — Bei diesem Ratespiel versucht der Ratende durch Fragen, die nur mit einem Ja oder Nein beantwortet werden, einen bestimmten Begriff herauszufinden. Seit Jahrzehnten lebt die Fernsehunterhaltung von immer neuen Variationen dieser einen bescheidenen Idee.
Auch diese Art des Problemlösens kann als Entscheidungsbaum dargestellt werden. Jede Frage entspricht einem Knotenpunkt, bei dem sich der Lösungsweg in eine Ja- und Nein-Richtung gabelt. Auf dieser Idee basiert das Programm RATE.LRN, das alles lernt, was man ihm eintippt, und Begriffe rät, die man sich ausdenkt.
Nach dem Start stellt das Programm die Standardfrage, ob es sich bei dem gesuchten Begriff um eine Sache handelt. Der Benutzer kann mit Ja oder Nein antworten, das Programm wird auf jeden Fall bekennen, das Gesuchte sei noch unbekannt. Es hat ja schließlich noch nichts gelernt.
Der Benutzer wird aufgefordert, den gesuchten Begriff einzutippen. Nun soll er die Frage eingeben, die zum Lösungswort führt. Schließlich ist noch die Antwort auf diese Frage zu benennen. Sie bestimmt den gesuchten Begriff endgültig. Nach diesen Eingaben ist das Programm in der Lage, den Begriff durch fortgesetztes Fragen zu ermitteln.
Offen gesagt, die »Intelligenz« dieses Programms beruht eher auf einem Taschenspielertrick. Bevor wir das Geheimnis jedoch Zeile für Zeile lüften, sollten Sie sich den Spaß gönnen:

Programm RATE.LRN

```
1 REM RATE.LRN
10 GOSUB 30010
20 GOSUB 20010
100 n=1
110 GOSUB 1000
120 GOSUB 10010
130 CLS:PRINT:PRINT "Neuer Durchgang":PR
INT:PRINT
140 GOTO 100
1000 PRINT f$(n);" ...":PRINT
1010 INPUT i$:PRINT
1020 i$=CHR$(ASC(i$))
1030 IF i$="J" OR i$="j" AND a(n)=0 GOTO
 2010
1040 IF i$="N" OR i$="n" AND b(n)=0 GOTO
 2010
1050 IF i$="J" OR i$="j" AND a(n)<0 THEN
 c=-a(n):GOTO 3010
```

Programm
RATE.LRN

```
1060 IF i$="N" OR i$="n" AND b(n)<0 THEN
 c=-b(n):GOTO 3010
1070 IF i$="J" OR i$="j" THEN n=a(n)
1080 IF i$="N" OR i$="n" THEN n=b(n)
1090 GOTO 1000
2000 REM > noch unbekannt <
2010 i=d+1:j=0
2020 GOSUB 5010
2030 RETURN
3000 REM > Antwort-Versuch <
3010 PRINT "Ist es ein ... ";o$(c):PRINT
3020 INPUT a$:PRINT:a$=CHR$(ASC(a$))
3030 IF a$="J" OR a$="j" THEN PRINT "Das
 wußte ich!":PRINT:PRINT:RETURN
4000 REM > Versuch falsch <
4010 i=f+1:j=-c
4020 GOSUB 5010
4030 RETURN
5000 REM > hinzulernen <
5010 PRINT "Das Gesuchte ist noch unbeka
nnt!":PRINT
5020 PRINT "Wie ist sein Name ...":PRINT
5030 INPUT a$:PRINT
5040 PRINT "Wie lautet die Frage,":PRINT
5050 PRINT "um das Gesuchte zu bestimmen
 ...":PRINT
5060 INPUT b$:PRINT
5070 o=o+1:o$(o)=a$:f=f+1:f$(f)=b$:d=d+1
5080 IF i$="J" OR i$="j" THEN a(n)=i
5090 a(d)=-o:b(d)=j
5100 IF i$="N" OR i$="n" THEN b(n)=i
5110 PRINT "Was ist die Antwort auf ..."
:PRINT
5120 PRINT b$:PRINT
5130 INPUT c$:PRINT:c$=CHR$(ASC(c$))
5140 IF c$="N" OR c$="n" THEN b(d)=-o:a(
d)=j
5150 RETURN
10000 REM > Tastaturabfrage <
10010 PRINT "Bitte drücken Sie eine Tast
e"
10020 IF INKEY$="" GOTO 10020
10030 RETURN
20000 REM > Bildschirmgestaltung <
20010 MODE 1
20020 LOCATE 4,3:PRINT"****************
*****************"
20030 LOCATE 4,4:PRINT"****************
*****************"
20040 LOCATE 4,5:PRINT"**
                             **"
20050 LOCATE 4,6:PRINT"**     A L L E S
L E R N E R     **"
20060 LOCATE 4,7:PRINT"**
                             **"
20070 LOCATE 4,8:PRINT"****************
*****************"
20080 LOCATE 4,9:PRINT"****************
*****************"
20090 j=999:DIM f$(j),a(j),b(j),o$(j)
```

Programm
RATE.LRN

```
20100 d=1:f=1
20110 f$(1)="IST ES EINE SACHE"
20120 LOCATE 7,25:GOSUB 10010
20130 LOCATE 7,25:PRINT "
                    ";
20140 LOCATE 1,15:PRINT "Ich lerne alles
, was Sie mir beibringen!"
20150 LOCATE 6,25:GOSUB 10010
20160 LOCATE 6,25:PRINT "
                    ";
20170 LOCATE 1,15:PRINT "Bitte beachten
Sie die Tastaturbelegung!"
20180 LOCATE 2,17:PRINT "Die Antworten J
a oder Nein können Sie"
20190 LOCATE 3,19:PRINT "beliebig groß o
der klein schreiben."
20200 LOCATE 6,25:GOSUB 10010
20210 MODE 1:RETURN
30000 REM > Umlaute <
30010 SYMBOL AFTER 91
30020 SYMBOL 91,66,24,60,102,126,102,102
,0
30030 SYMBOL 92,130,56,108,198,198,108,5
6,0
30040 SYMBOL 93,66,0,102,102,102,102,60,
0
30050 SYMBOL 123,68,0,120,12,124,204,118
,0
30060 SYMBOL 124,36,0,60,102,102,102,60,
0
30070 SYMBOL 125,36,0,102,102,102,102,62
,0
30080 SYMBOL 126,60,102,124,102,102,102,
108,96
30090 RETURN
65000 FOR j=0 TO d+1:PRINT j;": ";f$(j);
" / ";o$(j);" / ";a(j);b(j):NEXT j
```

Natürlich rät das Programm die Begriffe nicht so, wie ein menschlicher Spieler sie erraten würde. Ohne die Beschränkung auf einen speziellen Themenkreis ist heute wohl noch kein System denkbar, das einen Rateprozeß optimal simulieren kann. Das Raten erfordert nämlich nicht nur ein umfassendes Alltagswissen, sondern auch eine ordnende Hierarchie von Merkmalen für alle möglichen Objekte.
Der menschliche Rater wird, wenn er klug ist, von globalen Begriffen ausgehend das Spektrum der Möglichkeiten durch gezielte Fragen zielstrebig begrenzen. Dabei bringt ihn jede Antwort, ob Ja oder Nein, dem Ziel näher. Seine Fragen sind Filter oder Siebe, die irgendwann nur noch einen bestimmten — den gesuchten — Begriff zulassen.
Dieses Programm geht die diffizile Aufgabe des Begrifferatens recht plump an. Es ist entsprechend wenig lei-

Vom Begriff »Lernen« nicht zu trennen: Der Begriff »Speichern«

stungsfähig, auch wenn es beim ersten Ansehen verblüffen mag, daß es jeden einmal gelernten Begriff immer wieder errät. Dazu ist es allerdings nur in der Lage, weil es alle Benutzereingaben während einer Sitzung protokolliert und dabei Schritt für Schritt einen Entscheidungsbaum aufbaut. Mit jeder Eingabe wird eine weitere Verzweigung angelegt. Die Knotenpunkte sind durch die vom Benutzer definierten Fragen markiert.
Am Ende jedes Ja-Nein-Weges steht ein Suchbegriff. Das bedeutet aber, daß der erste, vom Benutzer eingegebene Begriff nach der zweiten Frage gefunden wird. Für das Finden von später eingegebenen Begriffen werden immer mehr Fragen benötigt. Eine weitere Beschränkung liegt darin, daß eine eindeutige Begriffs-Frage-Verbindung bestehen muß. Soll zum Beispiel das Objekt »Hase« durch die Frage: »Hat es Ohren?« bestimmt werden, kann die gleiche Frage nicht auch noch auf Hunde oder andere beohrte Objekte zielen. Mit anderen Worten: Das Programm hat nur einen begrenzten Wert; es dient der Klassifizierung von Gegenständen, die durch ein einziges Merkmal zu identifizieren sind.

Programmbeschreibung RATE.LRN

Das Programm RATE.LRN beginnt mit dem Aufruf von zwei Unterprogrammen. Ab **Zeile 30000** werden die deutschen Sonderzeichen definiert und, entsprechend dem internationalen ASCII-Standard, den Codes 91 bis 93 und 123 bis 126 zugeordnet. Ab **Programmzeile 20000** wird der Vorspann erzeugt, und die Variablen werden dimensioniert, in denen im Verlauf der Sitzung neues Wissen erfaßt wird. Die Feldvariable f$ nimmt die Fragen auf, die Werten zugeordnet sind, die als Index auf die in der Feldvariablen o$ gespeicherten Gegenstände zeigen. Die Feldvariable a nimmt den Indexwert auf, der bei einer Ja-Antwort des Benutzers eingesetzt werden soll, die Feldvariable b den, der für eine Nein-Eingabe vorgesehen ist. Da das Programm zu Beginn über eine Frage verfügen muß, wird für f$(1) ein Inhalt vorgegeben.
Die eigentliche Arbeit des Programms beginnt in **Zeile 100**. Sie besteht aus dem Aufruf des Unterprogramms ab **Zeile 1000** und endet mit der Tastaturabfrage in **Zeile 120**, die ebenfalls in einem Unterprogramm untergebracht ist.
In **Zeile 1000** wird mit f$(n) eine Frage ausgegeben. Da n anfangs auf 1 gesetzt wird, ist dies im ersten Durchgang die Frage: »Ist es eine Sache?«
Zeile 1010 nimmt die Benutzerantwort auf, und **Zeile 1020** reduziert sie auf den ersten Buchstaben. Da bis **Zeile 1090** nun noch IF-Abfragen folgen, die eine zulässige Eingabe (»J«, »j«, »N« oder »n«) voraussetzen, wird bei un-

Programmbeschreibung
RATE.LRN

korrekter Eingabe in **Zeile 1090** nach **Zeile 1000** zurückgesprungen, wo dieselbe Frage noch einmal ausgegeben wird.
Wenn die mit der Frage n verbundene Feldvariable a(n) (bei Ja-Antwort) oder b(n) (bei Nein-Antwort) den Wert 0 enthält, handelt es sich um ein noch unbekanntes Objekt. Das Programm verzweigt nach **Zeile 2010** und von dort weiter nach **Zeile 5010**, wo der Name des Objekts und die Frage, mit der es gefunden werden soll, vom Benutzer eingegeben werden müssen. Die Eingaben werden in den Feldvariablen o$ und f$ abgelegt.
Der Wert von o, der als Index in der Form o$(o) auf den Objektnamen in der Feldvariablen o$ weist, wird als negativer Wert abgelegt. Dies geschieht in der Feldvariablen a, wenn die Antwort auf die Frage zur Bestimmung des Objekts »Ja« lautet, in b, wenn sie »Nein« heißt. Der Index für a oder b entspricht dem Index, unter dem die zugehörige Frage im String-Array f$ abgelegt ist.
Zu jedem beliebigen Fragestring f$(n) halten die Variablen a(n) und b(n) also je einen Wert bereit. Einer von ihnen ist negativ. Er zeigt auf das gesuchte Antwortobjekt. Muß der Fragestring mit »Ja« beantwortet werden, um das Objekt zu bestimmen, wird der negative Indexwert in a(n), bei »Nein« in b(n) abgelegt. Die jeweils andere Antwortmöglichkeit bleibt frei: Sie ist 0, wenn noch kein Objekt bekannt ist, das durch diese Frage und die entsprechende Antwort bestimmt wird; oder sie hält einen positiven Wert, der als Index auf die nächste zu stellende Frage zeigt; oder sie hält auch einen negativen Wert, wenn auf eine zweite Antwort verwiesen wird. Es ist ja möglich, daß auf eine Frage mit Ja- und mit Nein-Antwort jeweils ein anderes Objekt bestimmt wird.
Nachdem das Programm auf diese Weise den Benutzer ausgefragt und seine Antworten gelernt hat, führt der RETURN-Sprung aus **Zeile 2030** nach **Programmzeile 110**, wo ein neuer Durchgang beginnt.
Findet das Programm in der **Zeile 1050** oder **1060** einen negativen Wert, so ist eine Antwort bekannt. Sie wird in **Zeile 3010** in Form einer Frage ausgegeben. Bestätigt der Benutzer die Antwort des Computers, ist das Fragespiel beendet. Verneint er jedoch, lernt das Programm auf die beschriebene Weise im Unterprogramm ab **Zeile 5010** die neue Antwort hinzu.
Findet das Programm in **Zeile 1070** oder **1080** einen positiven Wert, so nimmt es ihn als neuen Index und gibt nach dem Rücksprung von **Zeile 1090** nach **Zeile 1000** die entsprechende Frage aus. Das Austauschen des Index' und das Ausgeben einer neuen Frage geht so lange weiter, bis ein Indexwert 0 oder ein negativer Wert

gefunden wird. Im ersten Fall ist das gesuchte Objekt noch unbekannt, im zweiten handelt es sich um den Index auf eine Antwort.
Wie leicht dieses Quizprogramm zu verwirren ist, werden Sie herausfinden, wenn Sie ein wenig damit spielen. Wollen Sie sich den Inhalt des Computergedächtnisses anschauen, können Sie das Programm durch zweimaliges Drücken der <ESC>-Taste unterbrechen. Geben Sie danach den Befehl ein: GOTO 65000. Die Inhalte der Feldvariablen f$, o$, a und b erscheinen nun fortlaufend numeriert auf dem Bildschirm.

Auf ein Wort

Vor einigen Jahren kam der Vizepräsident einer Bostoner Computerfirma an einem Samstag in sein Büro, um liegengebliebene Arbeiten aufzuholen. Als er den Computer benutzen wollte, bemerkte er, daß die Anlage in Betrieb war. Er setzte sich an das Terminal und fragte den vermeintlichen anderen Benutzer: »Meinen Sie, ich könnte heute morgen den Computer benutzen?«
Die Antwort kam augenblicklich: »Warum fragen Sie?«
»Möglicherweise kann ich noch ein paar Geräte mehr verkaufen.«
»Warum sind Sie sich da nicht sicher?«
»Meinen Kunden ist das System noch nicht vorgeführt worden.«
»Bedeutet es so viel für Sie?«
Der Vizepräsident muß viel Geduld gehabt haben, denn er antwortete noch einmal: »Selbstverständlich!«
Als aber die Erwiderung kam: »Sind Sie sich da sicher?« wurde er skeptisch: »Sie wollen mich wohl auf den Arm nehmen?«
Bei der Gegenfrage: »Was heißt das, auf den Arm nehmen?« platzte ihm dann endgültig der Kragen: »Rufen Sie sofort 491 an!«
Natürlich rief niemand an, denn, ob versehentlich oder mit schelmischer Absicht, der Computer war nicht ausgeschaltet worden, und ein Programm namens ELIZA geisterte durch seine Schaltkreise. Dieses Programm hatte soeben den Turing-Test bestanden, nach dem ein Programm dann als intelligent eingestuft werden soll, wenn ein menschlicher Partner mit ihm kommuniziert, ohne zu bemerken, daß es sich nur um die Erwiderungen einer Maschine handelt. Und für diesen Beweis, daß man einen menschlichen Gesprächspartner auch mit sprachlichen Tricks vorgaukeln kann, hat der Computerwissenschaftler Joseph Weizenbaum ELIZA erschaffen.

Das Programm ELIZA: Was wie ein Psychotherapeut reagiert, ist noch lange keiner

Es (oder müssen wir sagen: sie?) reagiert etwa so wie ein Psychotherapeut in einem Erstgespräch, wenn er den Patienten lediglich zum Sprechen bringen will. Dazu werden die Aussagen des Benutzers in Fragen umgewandelt. Es werden, ausgelöst von bestimmten, in den Benutzereingaben enthaltenen Stichwörtern, Fragen gestellt oder Aussagen getroffen. Oder es werden einfach Phrasen ausgegeben, die den Benutzer zu weiteren Aussagen stimulieren. Auf diese Weise kann ein

endloser Austausch von Sätzen erfolgen, der in dem Maße neue Inhalte aufgreift, wie der menschliche Benutzer neue Inhalte einbringt und der sich in dem Maße wiederholt, wie sich Standardphrasen wiederholen müssen, weil das interne Lexikon des Programms begrenzt ist. Ein solcherart konzipiertes Dialogprogramm ist nichts weiter als ein nach den Regeln der Grammatik arrangiertes Echo.

Während die Idee zu diesem Programm für die kritische Intelligenz seines Programmierers spricht, kann das Programm selbst nicht als sonderlich intelligent bezeichnet werden. Es handelt sich um keinen praktischen Beitrag zur KI. Die Methode besteht darin, eine vom Benutzer eingegebene Zeichenfolge nach vorgegebenen Stichwörtern zu durchsuchen, in grammatische Elemente zu zerlegen und diese Elemente zu neuen Satzgebilden zusammenzusetzen. Eine Analyse der Inhalte und eine intelligente Reaktion darauf erfolgt nicht.

Ein grundlegendes Problem: Die inhaltliche Analyse menschlicher Sprache durch den Computer

Tatsächlich stellt die Aufgabe, Programme zu befähigen, die Inhalte sprachlicher Äußerungen zu erkennen, die Programmierer vor eine Menge Probleme. Jede menschliche Kommunikation basiert auf einer unüberschaubaren Menge von Informationen, die bei den menschlichen Kommunikationspartnern als bekannt vorausgesetzt werden können.

Stellen Sie sich vor, Sie rufen in einem Hotel an, um ein Zimmer zu bestellen. Nach den üblichen Fragen nach Termin, Dauer des Aufenthalts und Übernachtungspreis wird über die Ausstattung des Zimmers gesprochen. Sie wollen während Ihres Hotelaufenthaltes arbeiten und fragen deshalb, ob sich in dem Zimmer ein Schreibtisch befindet. Der Portier antwortet Ihnen, im Zimmer stehe zwar kein Schreibtisch, aber dafür sei ein großer Tisch vorhanden, an dem man bequem arbeiten könne. Ein alltäglicher Vorgang, der nicht weiter erwähnenswert wäre, wenn der Portier nicht durch einen Auskunfts-Computer ersetzt werden sollte.

Würde der Portier einfach nur der Wahrheit entsprechend antworten, das Zimmer sei nicht mit einem Schreibtisch ausgestattet, würden Sie es vielleicht nicht buchen und sich ein anderes Hotel suchen.

Was muß der Portier wissen, um die Auskunft geben zu können? Er muß den Begriff Schreibtisch kennen und schlußfolgern, daß Sie danach fragen, weil Sie arbeiten wollen. Da er weiß, daß das Zimmer nicht mit einem Schreibtisch ausgestattet ist, muß er nach einer Alternative suchen. Er weiß, daß man auch an anderen Tischen sitzend arbeiten kann. Um diese Alternative attraktiv

anbieten zu können, muß der Portier erkennen, daß die Frage nach einem Schreibtisch als Arbeitsplatz durch den Wunsch nach Bequemlichkeit motiviert ist. Nun kann er den Tisch mit den Worten »bequemer Arbeitsplatz« anbieten.

Wenn Sie sich vorstellen, wie viele verschiedene Anfragen buchender Gäste an einen Hotelportier gestellt werden können, wird deutlich, wie komplex ein Programm sein müßte, damit ein Anrufer, wenn nur die automatische Stimme ganz natürlich klingen würde, nicht erkennt, daß er von einem Rechner abgefertigt wird. Wenn Sie sich ferner überlegen, welch ein bescheiden winziger Zipfel menschlicher Alltagsgespräche die begrenzte Umgebung des Buchens von Hotelzimmern darstellt, können Sie ungefähr ahnen, wie weit Maschinen noch davon entfernt sind, sich adäquat mit uns zu unterhalten.

Nachdem Sie diese Zeilen gelesen haben, werden Sie, wie ich hoffe, das folgende Programm großmütig beurteilen: Es unterhält sich mit Ihnen beliebig lange und über beliebige Inhalte. Aber es ist nicht besonders redegewandt, und wenn Sie nicht auf sein Spezialthema eingehen und Ihre Sätze nicht ganz einfach aufbauen, ist die Unterhaltung nicht sehr abwechslungsreich.

LINDA: Ein Programm nicht nur für Depressive

DEPRESS.DIA arbeitet ähnlich wie ELIZA, nur ist sein Wortschatz ganz auf einen deprimierten Benutzer eingestellt. Das Programm stellt sich als Linda vor und bringt eine Begrüßung auf den Bildschirm. Nachdem es den Namen des Benutzers erfragt hat — nennen wir ihn oder sie »Lesa« —, eröffnet es das Gespräch. Die Unterhaltung kann dann etwa wie folgt verlaufen; die Benutzereingaben sind zur besseren Unterscheidung klein geschrieben, tatsächlich sollten sie in Großbuchstaben erfolgen; das Programm verzichtet auf Satzzeichen — die einzige Ausnahme von dieser Regel bildet das Fragezeichen hinter der Eingangsfrage.

WARUM BIST DU DEPRIMIERT LESA?
Ich verstehe die KI nicht
EINEM FABELHAFTEN MENSCHEN WIE DIR HÖRE ICH GERNE ZU BITTE ERZÄHLE MIR MEHR ÜBER DICH LESA
Ich brauche Hilfe bei der Einarbeitung
WIE MEINST DU KÖNNTE DIR GEHOLFEN WERDEN
Durch ein gutes Buch
ICH KANN DAS LEICHT NACHEMPFINDEN WORUNTER LEIDEST DU BESONDERS LESA
Es quält mich dass ich vieles nicht verstehe
WAS FÜR GEFÜHLE LÖST DAS BEI DIR NOCH AUS

Und so könnte das endlos weitergehen. Wenn sich der Benutzer an die Vorgabe hält, daß seine Aussagen depressiv gefärbt sein sollen, entsteht sogar meist eine ganz ordentliche Unterhaltung. Wer es aber darauf angelegt, wird Linda leicht so verwirren können, daß ihre Antworten kaum noch Sinn ergeben.

Programm DEPRESS.DIA

```
1 REM DEPRESS.DIA
10 MODE 1:RANDOMIZE TIME
20 GOSUB 60010
30 GOSUB 10010
40 FOR j=0 TO 17:FOR i=0 TO 3:READ c$(j,
i):NEXT i:NEXT j
50 FOR j=0 TO 6:FOR i=0 TO 1:READ f$(j,i
):NEXT i:NEXT j
60 FOR j=0 TO 29:FOR i=0 TO 2:READ g$(j,
i):NEXT i:NEXT j
70 FOR j=0 TO 4:READ r$(j):NEXT j
80 FOR j=0 TO 6:READ s$(j):NEXT j
100 PRINT:PRINT:PRINT:PEN 1:INPUT e$:PRI
NT:PRINT:PRINT:PEN 2
200 IF e$="ENDE" GOTO 30010
210 IF e$="JA" OR e$="NEIN" THEN GOSUB 2
0010:GOTO 100
300 f=0:a=0
310 FOR j=0 TO 17
320 f=INSTR(e$,c$(j,0))
340 IF f>0 THEN a=1:b=j
350 NEXT j
360 IF a=1 THEN GOSUB 21010:GOTO 100
400 a=0:FOR j=0 TO 24
410 f=LEN(g$(j,0))
420 IF LEFT$(e$,f)=g$(j,0) THEN a=1:b=j:
c=f
430 NEXT j
440 IF a=1 THEN GOSUB 23010:GOTO 100
500 a=0:FOR j=0 TO 6
510 f=LEN(f$(j,0))
520 IF LEFT$(e$,f)=f$(j,0) THEN a=1:b=j:
c=f
530 NEXT j
540 IF a=1 THEN GOSUB 22010:GOTO 100
600 q=INT(RND(1)*7)
610 PRINT s$(q)+" "+p$
620 GOTO 100
10000 REM > Vorspann <
10010 DIM c$(17,3),f$(6,1),g$(29,2),r$(4
),s$(6):k$="KEIN":t$="NICHT":w$="WARUM"
10020 INK 0,0:INK 1,11:INK 2,20:INK 3,19
10030 BORDER 0:CLS
10040 PEN 2
10050 PRINT"GUTEN TAG":GOSUB 11000
10060 PRINT"MEIN NAME IST LINDA":GOSUB 1
1000
10070 PRINT"WENN DU MIT MIR ÜBER DEINEN
KUMMER SPRECHEN WILLST, WERDE ICH DIR GE
RNE ZUHÖREN":GOSUB 11000
```

Programm
DEPRESS.DIA

```
10080 PRINT"ICH HABE VIEL ZEIT FUR DICH,
 SO VIEL ZEIT, WIE DU WILLST":GOSUB 1100
0
10090 PRINT"WIE IST DEIN NAME":PRINT:PRI
NT:PRINT
10100 PRINT"BITTE SCHREIBE NUR MIT GROSS
EN BUCHSTABEN. AM BESTEN DRUCKST DU DIE
TASTE <CAPS LOCK>":PRINT
10105 PRINT"UND BEENDE JEDE EINGABE MIT
<RETURN>":PRINT:PRINT:PRINT
10110 PEN 1
10120 INPUT P$
10130 PEN 2:PRINT:PRINT:PRINT
10140 PRINT"ICH FREUNE MICH, DICH KENNEN
ZULERNEN, ":PRINT P$:PRINT"MACH ES DIR B
EQUEM UND DANN FANGEN WIR AN":GOSUB 1100
0
10200 WINDOW #1,1,40,1,8:WINDOW #0,1,40,
9,25
10210 CLS #1:CLS:PEN #1,3
10220 PRINT #1,"> BITTE GIB NUR GROSSBUC
HSTABEN EIN":PRINT #1
10230 PRINT #1,"> GIB KURZE, EINFACHE SA
TZE EIN, DAMIT     ICH DICH GUT VERSTEHEN
 KANN":PRINT #1
10240 PRINT #1,"> VERZICHTE AUF =ALLE= S
ATZZEICHEN"
10250 PRINT #1,"------------------------
----------------";
10260 CLS:PEN 2:PRINT"WARUM BIS DU DEPRI
MIERT, ";P$
10280 PRINT:PRINT:PRINT
10290 RETURN
11000 LOCATE 15,20:PEN 3:PRINT"WEITER MI
T"
11010 LOCATE 12,21:PRINT"BELIEBIGER TAST
E"
11020 IF INKEY$="" GOTO 11020
11030 CLS:PEN 2:RETURN
20000 REM > Nachfrage <
20010 q=INT(RND(1)*4)+1:PEN 2
20020 ON q GOTO 20030,20040,20050,20060
20030 PRINT"BITTE ERZÄHL MIR MEHR DARÜBE
R, DAMIT ICH MIR EIN BESSERES BILD MACHE
N KANN":GOTO 20070
20040 PRINT"KANNST DU DAS ETWAS NÄHER ER
LÄUTERN":GOTO 20070
20050 PRINT"DAS INTERESSIERT MICH, BITTE
 ERZÄHL WEITER":GOTO 20070
20060 PRINT"DU MUSST SCHON ETWAS MEHR ER
ZÄHLEN, WENN ICH DICH VERSTEHEN SOLL"
20070 RETURN
21000 REM > Schlüsselwörter-Antwort <
21010 q=INT(RND(1)*3)+1:PEN 2
21020 a$=c$(b,q):PEN 2
21030 PRINT a$
21040 RETURN
22000 REM > Ich-Sätze <
22010 x$=MID$(e$,c+2,LEN(e$)-c+1)
22020 a$=f$(b,1)+" "+x$+", "+p$
```

Programm
DEPRESS.DIA

```
22030 PRINT a$
22040 RETURN
23000 REM > Ich-Prädikat-Sätze <
23010 IF MID$(e$,c+1,5)=t$ THEN GOSUB 23
510:RETURN
23020 IF MID$(e$,c+1,4)=k$ THEN GOSUB 23
610:RETURN
23030 x$=RIGHT$(e$,LEN(e$)-c-1)
23040 a$=g$(b,1)+" WIRKLICH "+x$
23050 PRINT a$
23060 RETURN
23500 REM > verneinte Sätze <
23510 x$=RIGHT$(e$,LEN(e$)-c-7)
23520 q=INT(RND(1)*5
23530 a$=r$(q)+" "+g$(b,2)+" "+t$+" "+x$
23540 PRINT a$
23550 RETURN
23600 REM > ver-kein-te Sätze <
23610 x$=RIGHT$(e$,LEN(e$)-c-2)
23620 a$=w$+" "+g$(b,1)+" "+x$+" "+p$
23630 PRINT a$
23640 RETURN
30000 REM > Ende <
30010 PRINT:PRINT:PRINT"VIELEN DANK ";P$
30020 PRINT:PRINT:PRINT"DU KANNST JEDERZ
EIT WIEDERKOMMEN"
30030 PRINT:PRINT:PRINT"ICH HABE MICH GE
RNE MIT DIR UNTERHALTEN"
30040 PRINT:PRINT:PRINT"ICH MAG DICH!"
30050 FOR J=0 TO 1000:NEXT J
30060 MODE 1:END
50000 DATA SINNLOS
50010 DATA MANCHMAL SIEHT ALLES NUR SINN
LOS AUS
50020 DATA WARUM VERBOHRST DU DICH SO IN
 DIE FRAGE NACH DEM SINN
50030 DATA KANNST DU NICHT AUCH GLÜCKLIC
H SEIN OHNE DEN SINN ZU VERSTEHEN
50040 DATA TRAU
50050 DATA AUS TRAUER KANN MAN AUCH KRAF
T SCHÖPFEN MEINST DU NICHT
50060 DATA WARUM FÄLLT ES DIR SO SCHWER
ZU TRAUERN
50070 DATA WAS GEHT IN DIR VOR WENN DU T
RAURIG BIST
50080 DATA KUMMER
50090 DATA WARUM MACHST DU ES DIR SO SCH
WER
50100 DATA DU KANNST NICHT FUR ALLES VER
ANTWORTLICH SEIN
50110 DATA WARUM BELASTEST DU DICH DAMIT
50120 DATA SORG
50130 DATA WARUM VERSTEIGST DU DICH IN Ä
NGSTE
50140 DATA ERZÄHLE GENAUER WAS DICH DARA
N SO BEDRÜCKT
50150 DATA WAS BESORGT DICH AM MEISTEN
50160 DATA ZWECKLOS
50170 DATA WARUM HÄLST DU DAS FÜR ZWECKL
OS
```

Programm
DEPRESS.DIA

```
50175 DATA MACHST DU DIR OFT SORGEN
50180 DATA WARUM WILLST DU ES NICHT NOCH
 EINMAL VERSUCHEN
50190 DATA VERSAG
50200 DATA DU WILLST NICHT SAGEN DU HATT
EST NOCH NIE ETWAS GELEISTET ODER
50210 DATA MAN KANN NICHT IMMER ERFOLGRE
ICH SEIN WARUM BIST DU SO HART ZU DIR
50220 DATA MEIST DU WIRKLICH DU HATTEST
VERSAGT
50230 DATA HILF
50240 DATA WO BRAUCHST DU AM MEISTEN HIL
FE
50250 DATA WIE MEINST DU KÖNNTE DIR GEHO
LFEN WERDEN
50260 DATA WAS WÜRDE DEINE NOT LINDERN
50270 DATA HELFEN
50290 DATA DU MUSST DIR ABER AUCH HELFEN
 LASSEN
50295 DATA AN WELCHE ART HILFE DENKST DU
 DABEI?
50300 DATA MANCHMAL KOMMT HILFE GANZ UNE
RWARTET
50310 DATA LEID
50320 DATA VERSTECKT SICH HINTER DEINEM
SCHMERZ VIELLEICHT NOCH ETWAS ANDERES
50330 DATA WAS BEREITET DIR DEN GROSSTEN
 KUMMER
50340 DATA DAS LEIDEN IST AUCH EINE ZEIT
 DER BESINNUNG UND SOMIT AUCH HILDFREICH
50350 DATA NIEDERGESCHLAGEN
50360 DATA HAST DU DICH ÜBERANSTRENGT
50370 DATA FALSCHE ERNÄHRUNG UND KÖRPERL
ICHE MANGELERSCHEINUNGEN KÖNNEN SEELISCH
E VERÄNDERUNGEN VERURSACHEN
50380 DATA DENKST DU DU BRAUCHST MEHR RU
HE
50390 DATA VERLUST
50400 DATA SICHER JEDER VERLUST IST SCHM
ERZHAFT ABER WIE KANN ES JETZT WEITERGEH
EN
50410 DATA OFT MUSS MAN ETWAS VERLIEREN
UM ETWAS NEUES GEWINNEN ZU KÖNNEN WAS ME
INST DU
50420 DATA MANCHMAL IST EIN VERLUST AUCH
 EINE BEFREIUNG
50430 DATA DRUCK
50440 DATA SETZT DU DICH VIELLEICHT AUCH
 SELBST UNTER DRUCK
50450 DATA WIE WIRKT SICH DER DRUCK AUF
DICH AUS
50460 DATA WIE REAGIERST DU AUF DEN DRUC
K
50470 DATA KRANK
50480 DATA KRANKHEIT IST OFT DER AUSDRUC
K SEELISCHER KONFLIKTE
50490 DATA KRANK SEIN IST AUCH EINE CHAN
CE MENSCHLICHE ERFAHRUNGEN ZU MACHEN
50500 DATA WIE KAM ES ZU DIESER KRANKHEI
T?
```

Programm
DEPRESS.DIA

```
50510 DATA TOD
50520 DATA ES BRINGT DICH NICHT WEITER V
OR DEM UNVERMEIDLICHEN ANGST ZU HABEN
50530 DATA WAS GENAU BEUNRUHIGT DICH BEI
 DEM GEDANKEN AN DEN TOD
50540 DATA VERSUCHST DU VIELLEICHT DAS M
ITLEID DEINER MITMENSCHEN ZU GEWINNEN
50550 DATA TOT
50560 DATA WAS BEUNRUHIGT DICH AN DIESER
 VORSTELLUNG
50570 DATA WELCHE VORSTELLUNGEN VERBINDE
ST DU MIT DEM TOD
50580 DATA HAST DU ANGST VOR DEM TOD
50590 DATA QUAL
50600 DATA GLAUBST DU WIRKLICH DASS ES S
O UNERTRÄGLICH IST
50610 DATA WENN MAN MIT SEINEM SCHMERZ A
LLEIN IST ERSCHEINT ER OFT VIEL GRÖSSER
50620 DATA DU SOLLTEST DICH VORÜBERGEHEN
D GANZ ANDEREN DINGEN ZUWENDEN UM ABSTAN
D ZU GEWINNEN
50630 DATA LAST
50640 DATA WENN ES SO UNERTRÄGLICH FÜR D
ICH IST SOLLTEST DU DANN NICHT EINE VERÄ
NDERUNG ANSTREBEN
50650 DATA DIE SITUATION WIRD DOCH AUCH
EINMAL VORÜBERGEHEN ODER
50660 DATA DU SOLLTEST DIR VIELLEICHT EI
NEN ERHOLUNGSURLAUB GÖNNEN
50670 DATA QUÄL
50680 DATA ES IST DOCH KEINE LÖSUNG WENN
 DU DIR SELBSTVORWÜRFE MACHST
50690 DATA WAS IST DARAN BESONDERS SCHLI
MM FÜR DICH
50700 DATA WAS FÜR GEFÜHLE LÖST DAS BEI
DIR NOCH AUS
51000 DATA ICH MUSS,NIEMAND MUSS WIRKLIC
H
51010 DATA ICH HÄTTE,WARUM HÄTTEST DU DE
NN
51020 DATA ICH WILL NICHT,KÖNNTEST DU DE
NN
51030 DATA ICH KANN NICHT,WILLST DU DENN
 ÜBERHAUPT
51040 DATA ICH DENKE,WARUM DENKST DU
51050 DATA ICH MEINE,WIESO MEINST DU
51060 DATA ICH FÜHLE MICH,AUS WELCHEM GR
UND FÜHLST DU DICH
52000 DATA ICH BIN,BIST DU,DU BIST
52010 DATA DU BIST,BIN ICH,ICH BIN
52020 DATA ES IST,IST ES,ES IST
52030 DATA WIR SIND,SEID IHR,IHR SEID
52040 DATA SIE SIND,SIND SIE,SIE SIND
52060 DATA ICH WAR,WARST DU,DU WARST
52070 DATA DU WARST,WAR ICH,ICH WAR
52080 DATA ES WAR.WAR ES,ES WAR
52090 DATA WIR WAREN,WART IHT,IHR WART
52100 DATA SIE WAREN,WAREN SIE,SIE WAREN
52110 DATA ICH WÜRDE,WÜRDEST DU,DU WÜRDE
ST
```

Programm
DEPRESS.DIA

```
52120 DATA DU WÜRDEST,WÜRDE ICH, ICH WÜR
DE
52130 DATA ES WÜRDE,WÜRDE ES,ES WÜRDE
52140 DATA WIR WÜRDEN,WÜRDET IHR,IHR WÜR
DET
52150 DATA SIE WÜRDEN,WÜRDEN SIE,SIE WÜR
DEN
52160 DATA ICH HABE,HAST DU,DU HAST
52170 DATA DU HAST,HABE ICH,ICH HABE
52180 DATA ES HAT,HAT ES,ES HAT
52190 DATA WIR HABEN,HABT IHR,IHR HABT
52200 DATA SIE HABEN,HABEN SIE,SIE HABEN
52210 DATA ICH HATTE,HATTEST DU,DU HATTE
ST
52220 DATA DU HATTEST,HATTE ICH,ICH HATT
E
52230 DATA ES HATTE,HATTE ES,ES HATTE
52240 DATA WIR HATTEN,HATTET IHR,IHR HAT
TET
52250 DATA SIE HATTEN,HATTEN SIE,SIE HAT
TEN
52260 DATA ICH WERDE,WIRST DU,DU WIRST
52270 DATA DU WIRST,WERDE ICH,ICH WERDE
52280 DATA ES WIRD,WIRD ES,ES WIRD
52290 DATA WIR WERDEN,WERDET IHR,IHR WER
DET
52300 DATA SIE WERDEN,WERDEN SIE,SIE WER
DEN
53000 DATA ICH MUSS DIR WIDERSRECHEN
53010 DATA DAS STIMMT NICHT
53020 DATA GUT ABER ICH FINDE
53030 DATA TROTZDEM MEINE ICH
53040 DATA DAS BILDEST DU DIR EIN DENN
53050 DATA WAS MACHT DICH DARAN SO UNGLÜ
CKLICH
53060 DATA EINEM FABELHAFTEN MENSCHEN WI
E DIR HÖRE ICH GERNE ZU BITTE ERZÄHLE ME
HR ÜBER DICH
53070 DATA AUF DEINE WEISE HAST DU SCHON
 RECHT BITTE ERZÄHLE WEITER DAMIT ICH DI
CH NOCH BESSER VERSTEHEN KANN
53080 DATA ICH KANN DAS GUT NACHEMPFINDE
N WORUNTER LEIDEST DU BESONDERS
53090 DATA WELCHE GEDANKEN MACHEN DICH B
ESONDERS MUTLOS
53100 DATA WARUM KANNST DU DICH NICHT AN
 DEN SCHÖNEN SEITEN DES LEBENS ERFREUEN
53110 DATA ICH MAG DICH DU KANNST GANZ O
FFEN ZU MIR SPRECHEN WOVOR HAST DU ANGST
60000 REM > Umlaute <
60010 SYMBOL AFTER 91
60020 SYMBOL 91,66,24,60,102,126,102,102
,0
60030 SYMBOL 92,130,56,108,198,198,108,5
6,0
60040 SYMBOL 93,66,0,102,102,102,102,60,
0
60050 RETURN
```

Programmbeschreibung DEPRESS.DIA

Die relative Länge des Programms ist auf den Umfang des internen Lexikons zurückzuführen, das in Form von Data-Zeilen die zweite Hälfte des Listings einnimmt. Dennoch ist DEPRESS.DIA nicht besonders wortgewaltig, und man könnte endlos weitere Daten für die Gestaltung von Lindas Antworten anfügen, indem man die Data-Zeilen entsprechend ergänzt. Natürlich müssen dann auch die verwendeten String-Arrays größer dimensioniert und alle Programmteile geändert werden, die sich auf die Menge der Daten beziehen.

In **Zeile 10** wird der Modus für die Bildschirmausgabe bestimmt und der Zufallszahlengenerator initialisiert.

Zeile 20: Im Unterprogramm ab **Programmzeile 60000** werden die deutschen Umlaute Ä, Ö und Ü auf die ASCII-Codes 91 bis 93 gelegt. Da der Benutzer aufgefordert wird, nur Großbuchstaben zu verwenden, entfallen die entsprechenden Kleinbuchstaben. Die Beschränkung auf Großbuchstaben und der Verzicht auf Satzzeichen vereinfacht die Programmstruktur.

In **Zeile 30** wird das Unterprogramm ab **Zeile 10000** aufgerufen. Dort werden in **Zeile 10010** Variablen dimensioniert, die das in den **Data-Zeilen** (ab **Zeile 50000**) abgelegte Vokabular von Linda aufnehmen sollen, und drei Strings definiert.

In den **Zeilen 10020 bis 10080** werden die Parameter für die Bildschirmgrafik festgelegt und ein Vorspann ausgegeben, mit dem sich das Programm vorstellt. Der Benutzer blättert durch die Bildschirmseiten, indem er eine beliebige Taste drückt. Die Abfrage der Tastatur erfolgt in dem bei **Zeile 11000** beginnenden Unterprogramm.

In **Zeile 10090** wird der Name des Benutzers erfragt und in **Zeile 10120** der Variablen p$ zugeordnet. Danach wird die Bildschirmgestaltung geändert. Eine ständige Anzeige erinnert den Benutzer daran, nur in Großbuchstaben zu schreiben, auf Satzzeichen zu verzichten und kurze, einfache Sätze zu verwenden. Dann eröffnet Linda das Programm mit der Frage: »Warum bis du deprimiert <Name>?« In **Zeile 10290** erfolgt der Rücksprung ins Hauptprogramm.

Dort werden von **Zeile 40 bis 80** die Textbausteine aus den Data-Zeilen in die verschiedenen Variablen eingelesen. Dieser Vorgang ist abgeschlossen, bevor der Benutzer seine Antwort auf die Eröffnungsfrage eingegeben hat, die von **Zeile 100** in die Variable e$ aufgenommen wird.

Zeile 200: Gibt der Benutzer ENDE ein, verabschiedet sich Linda im Unterprogramm ab **Zeile 30000** mit ein paar Artigkeiten.

Programmbeschreibung DEPRESS.DIA

Lautet die Antwort des Benutzers nur lapidar JA oder NEIN, sucht Linda im Unterprogramm ab **Zeile 20000** per Zufall einen von vier Sätzen aus, die den Benutzer auffordern, mehr zu erzählen.
Handelt es sich um eine längere Antwort, sucht sie die Benutzereingabe nach bestimmten Schlüsselwörtern durch. Zuerst werden in **Zeile 300** die beiden Hilfsvariablen a und f auf 0 gesetzt. Linda weiß auf achtzehn Schlüsselwörter, wie »sinnlos«, »trau« oder »quäl«, je drei Antworten. Dieses Vokabular ist in der Feldvariablen c$ abgelegt, wobei c$(j,0) das Schlüsselwort enthält und c$(j,1 bis 3) die Antworten darauf.
Die FOR-NEXT-Schleife von **Zeile 310 bis 350** geht in j die Werte 0 bis 17 durch und findet damit in **Zeile 320** das jeweilige Schlüsselwort c$(j,0). Ist c$(j,0) in der Benutzereingabe e$ enthalten, übergibt die Funktion INSTR an die Variable f die Stelle aus e$, an der c$(j,0) beginnt. Ist also in **Zeile 340** f größer als 0, ist das Schlüsselwort in der Eingabe enthalten. Dieser Umstand wird registriert, indem a den Wert 1 erhält. Die Variable b übernimmt mit dem Wert von j den Index auf das gerade bearbeitete Schlüsselwort.
Zeile 360: Wenn a den Wert 1 hat, also ein Schlüsselwort in der Eingabe enthalten ist, wird hier das Unterprogramm ab **Zeile 21010** aufgerufen.
In **Zeile 21010** bekommt q einen Zufallswert von 1 bis 3. Lindas Antwort a$ entspricht der zufällig gewählten Antwort q auf das Schlüsselwort c$(b,0), das in **Zeile 340** gefunden wurde. Die Variable c$(b,q) enthält demnach Lindas Antwort.
Nachdem a$ in **Zeile 21030** auf den Bildschirm geschrieben wurde, erfolgt der Rücksprung ins Hauptprogramm, wo mit GOTO 100 die nächste Benutzereingabe erwartet wird.
Hat das Programm kein Stichwort gefunden, sucht es in der nächsten Phase nach bestimmten Satzanfängen wie »ich bin« oder »sie werden«. Das Program kennt 25 solcher Subjekt-Prädikat-Kombinationen. Sie sind in der Feldvariablen g$ abgelegt, wobei g$(j,0) den Subjekt-Prädikat-String enthält und g$(j,1 und 2) die Antwort-Strings.
Die FOR-NEXT-Schleife in den **Zeilen 400 bis 430** geht mit j alle 25 Strings einzeln durch. In **Zeile 510** wird die Länge des gerade bearbeiteten Subjekt-Prädikat-Strings ermittelt und der Variablen f zugeordnet. Wenn das erste bis f-te Zeichen der Benutzereingabe e$ mit dem gerade bearbeiteten f$(j,0) identisch ist, hat der Benutzer eine bekannte Formulierung verwendet. Das Programm registriert diesen Fakt in den Variablen a

Programmbeschreibung DEPRESS.DIA

und b. Außerdem merkt es sich in c die Länge f dieses Stringteils.
Wurde eine bekannte Subjekt-Prädikat-Kombination gefunden, hat a den Wert 1, und **Zeile 540** verzweigt in das Unterprogramm ab **Zeile 23010**. Dort wird überprüft, ob auf die gefundene Subjekt-Prädikat-Kombination das Wort »nicht« folgt, das in t$ enthalten ist. Trifft diese Bedingung zu, wird das Unterprogramm ab **Zeile 23510** aufgerufen. Andernfalls wird in **Zeile 23020** überprüft, ob k$=»kein« folgt und entsprechend nach **Zeile 23610** verzweigt. Trifft auch diese Bedingung nicht zu, wird die Bearbeitung mit **Zeile 23030** fortgesetzt.
Wurde weder ein »nicht« noch ein »kein« gefunden, wird die Benutzereingabe in zwei Portionen zerlegt. Lautet sie zum Beispiel: »ich bin so müde«, wird der bekannte Teil, »ich bin«, abgetrennt und der Reststring, »so müde«, der Variablen x$ zugeordnet. Lindas Antwort a$ entsteht nun in **Zeile 23040** durch Verkettung der Antwortelemente, die in g$ unter dem Index (b,1) bereitgehalten werden. Im genannten Beispiel wäre das: »bis du«, der String »wirklich« und der abgetrennte Stringrest x$ »so müde«. So bekommt a$ den Wert: »bist du wirklich so müde«. **Zeile 23050** bringt diese Antwort auf den Bildschirm. Danach erfolgt der Rücksprung ins Hauptprogramm, wo mit GOTO 100 die nächste Benutzereingabe erwartet wird.
Hat der Benutzer einen Satz mit »nicht« eingegeben, wird das Unterprogramm ab **Zeile 23500** aufgerufen. Hier wird, wie oben beschrieben, der Eingabestring geteilt und die hintere Hälfte der Variablen x$ zugeordnet. Bei Eingabe von: »ich bin nicht so müde«, wäre das wiederum »so müde«. Die **Zeile 23520** ordnet der Variablen q einen Zufallswert von 0 bis 4 zu, der in **Zeile 23530** als Index verwendet wird, um eine der fünf in r$ abgelegten Widerspruchsphrasen zu bestimmen. Diese können zum Beispiel lauten: »ich muß dir widersprechen«, oder »das bildest du dir ein denn«.
Für den Anfangsteil der Benutzereingabe ist in g$(b,2) die zu einem verneinten Satz passende Formulierung abgelegt, im Beispiel wäre das: »du bist«. Die Variable t$ enthält das Verneinungswort »nicht«. Durch entsprechende Verkettung kommt Lindas Antwort zustande: »ich muß dir widersprechen« + »du bist« + »nicht« + »so müde«.
Zeile 23540 bringt diese Antwort auf die Mattscheibe, und in **Zeile 23550** erfolgt der Rücksprung ins Hauptprogramm, wo mit GOTO 100 die nächste Benutzereingabe erwartet wird.

Programmbeschreibung
DEPRESS.DIA

War der Satz mit »kein« verneint, erfolgt ab **Zeile 23610** eine ähnliche Behandlung. Die Antwort entsteht durch Verkettung von w$=»warum«, dem Erwiderungsbaustein g$(b,l) (»bist du«), dem Reststring x$ (»so müde«) und dem Namen des Benutzers p$. Nach dem Rücksprung ins Hauptprogramm wird nun die nächste Eingabe erwartet.
Hat die Durchsicht der Benutzereingabe nach den beschriebenen Mustern zu keiner Antwort geführt, läuft das Programm ab **Zeile 500** weiter. Hier wird festgestellt, ob sie mit einer Wendung wie »ich muß«, »ich hätte« oder »ich fühle mich« beginnt. Die FOR-NEXT-Schleife in den **Zeilen 500 bis 530** übernimmt diese Aufgabe und führt gegebenenfalls in das Unterprogramm ab **Zeile 22010**.
Dort wird wiederum der erkannte Teil abgetrennt (**Zeile 22010**). Durch Verkettung der entsprechenden Antwortphrase f$(b,l), des Reststrings x$ und des Benutzernamens p$ (**Zeile 22020**) wird die Antwort zusammengebaut und ausgegeben (**Zeile 22030**).
Hat das Programm immer noch keinen verwertbaren Inhalt in der Benutzereingabe e$ gefunden, wird die Bearbeitung mit **Zeile 600** fortgesetzt. Die Variable q erhält einen Zufallswert von 0 bis 6, der als Index in **Zeile 610** aus der Feldvariablen s$ eine von sieben unverbindlichen Reaktionen bestimmt, zum Beispiel: »einem fabelhaften Menschen wie dir höre ich gerne zu bitte erzähle mehr darüber«. So wird der Ball dem Benutzer zurückgespielt.
Nach diesem Muster können Programme für Dialoge über verschiedenste Themenbereiche geschrieben werden. Die aufwendigste Arbeit ist dabei die Analyse der wichtigsten Schlüsselbegriffe und Redewendungen.

Expertensysteme

Es wird viel über sie geredet, und wer mitreden will, muß sich mit Expertensystemen auseinandersetzen. Oft hat man aber den Eindruck, daß nicht jeder, der darüber redet, eine klare Vorstellung davon hat, was ein Expertensystem überhaupt ist oder sein sollte: ein System für Experten oder von Experten?

Expertensysteme unterscheiden sich von herkömmlichen Datenbanken durch ihre »intelligente« Struktur. Sie manifestiert sich in spezieller Hardware oder nur in Programmen und speichert die fachliche Kompetenz von Experten in Form von Sach- und Erfahrungswissen.

Dabei werden nicht nur Daten angehäuft, sondern auch nach Regeln strukturiert, verknüpft, klassifiziert, bewertet. Expertensysteme sind außerdem imstande, heuristische Methoden und vages Wissen einzubeziehen und nach den vorgegebenen Regeln aus dem vorhandenen Datenmaterial selbständig Schlüsse zu ziehen, das heißt, Problemlösungen anzubieten.

Expertensysteme erlauben interaktives Arbeiten

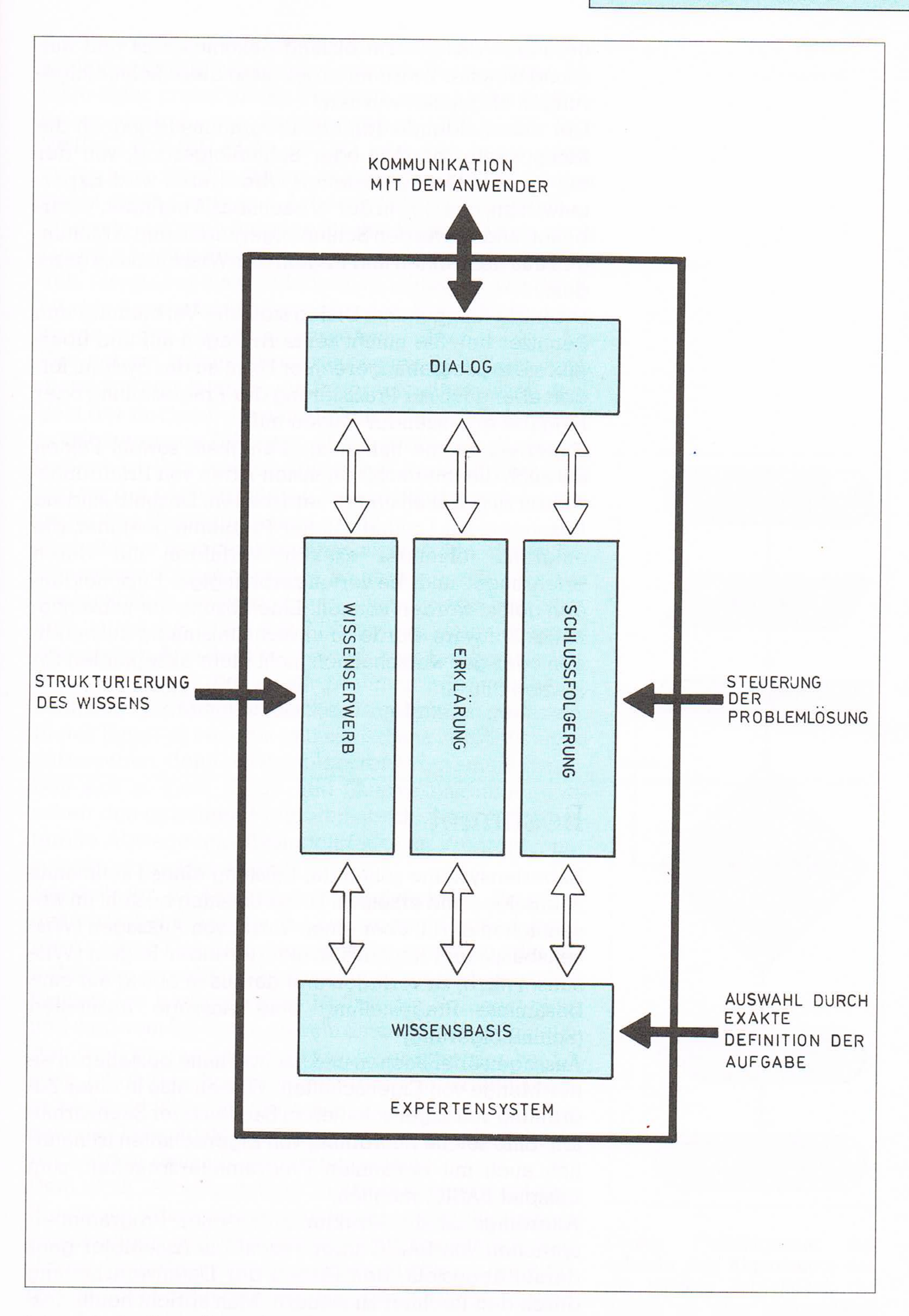
Darüber hinaus können sie an jeder Stelle des Lösungsprozesses Auskunft darüber geben, welche Annahme sie gerade verfolgen, warum sie einen eingeschlagenen Lösungsweg gewählt haben, zu welchen Schlußfolgerungen sie bereits gelangt sind, und warum sie zu diesen Schlußfolgerungen kamen. Auf diese Weise assistieren sie bei der Problemanalyse.

Ein Expertensystem setzt sich aus fünf Hauptkomponenten zusammen.

Der Grundstock ist die **Wissensbasis**, in der das Expertenwissen gespeichert und zum Beispiel in Form von Fakten und Regeln dargestellt ist. Um eine Abgrenzung vornehmen zu können, welche Wissenselemente in das System eingegeben werden müssen, ist eine exakte Definition der Aufgabe des bestimmten Expertensystems notwendig.

Die Komponente **Wissenserwerb** stellt Instrumente zur Verfügung, mit denen Expertenwissen in der Wissensbasis abgelegt werden kann. Dazu kann zum Beispiel eine Sprachstruktur gehören, durch die das Wissen logisch eindeutig formuliert werden kann, eine Hierarchie von Variablen, mit denen die Wissenselemente bewertet werden, ein Algorithmus, der Faktenlücken aufspürt und anderes mehr.

Die Komponente **Erklärung** gibt Auskunft darüber, welcher Lösungsweg verfolgt wird, zu welchen Schlußfol-



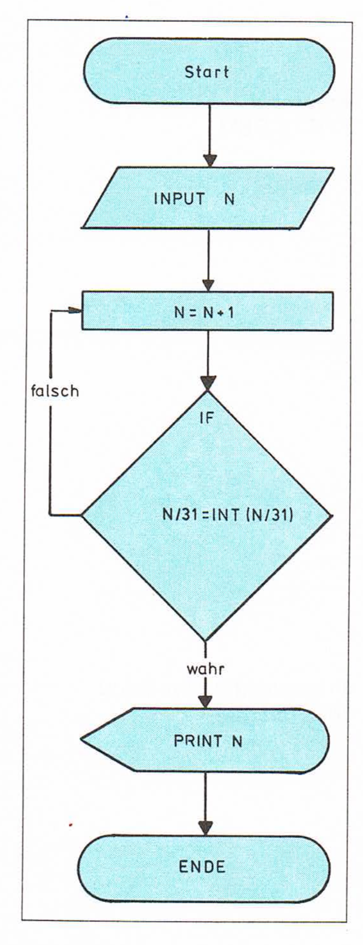
Schematische Darstellung eines Expertensystems

gerungen das System bislang gekommen ist und aufgrund welcher Zwischenergebnisse diese Schlußfolgerungen abgeleitet wurden.
Der entscheidende Teil des Programms ist jedoch die Komponente **Inferenz** oder Schlußfolgerung, von der die Problemlösung gesteuert wird. Dabei wird Expertenwissen, das sich in der Wissensbasis befindet, verarbeitet, und es werden Schlußfolgerungen und Ableitungen aus den Fakten und Regeln der Wissensbasis gezogen.
Die letzte Komponente **Dialog** stellt die Verbindung zum Benutzer her. Sie nimmt seine Anfragen auf und übergibt sie in programmgerechter Form an das System, fordert aber auch zur Präzisierung der Fragestellung oder Eingabe ergänzender Fakten auf.
Expertensysteme haben die Fähigkeit, sowohl Fakten als auch die unterschiedlichsten Arten von Erfahrungswissen zu erfassen und zu verarbeiten. Deshalb sind sie besonders zur Lösung solcher Probleme geeignet, die aufgrund fehlender exakter Verfahren nur durch erfahrungs- und bewertungsabhängige Entscheidungen gelöst werden können. Eine Lösung mit konventioneller Software würde zu unverhältnismäßig aufwendigen oder gar wirtschaftlich nicht mehr akzeptablen Systemen führen.

Bestimmt

Expertensysteme sollen die Leistung eines Fachmanns nachbilden und ersetzen. Diese Leistung besteht im wesentlichen darin, über einen Vorrat von Aussagen (Wissensbasis) und Kenntnis strukturierender Regeln (Wissenserwerb) zu verfügen und daraus in bezug auf eine bestimmte Fragestellung eine Aussage abzuleiten (Schlußfolgerung).
Aussagen über Sachen und Sachverhalte bestehen in einer Menge von Eigenschaften, Wissen also in einer Zuordnung von Eigenschaften zu Sachen oder Sachverhalten. Eine solche Zuordnung von Eigenschaften ist natürlich auch mit bekannten Programmiersprachen, zum Beispiel BASIC, möglich.
Allerdings ist die Struktur klassischer Programmiersprachen von BASIC über Pascal bis Assembler ganz darauf abgestellt, den Prozeß der Datenverarbeitung durch den Rechner zu steuern. Man spricht heute deshalb von prozeduralen Sprachen. Das Programm muß

die Verarbeitung der Daten vollkommen strukturieren, wenn das Ergebnis fehlerfrei sein soll. Die zentralen Befehle einer prozeduralen Programmiersprache funktionieren deshalb wie Weichensteller: FOR-NEXT, DO-WHILE, IF-THEN-ELSE, GOTO usw. Das heute allgemein vom Struktugramm abgelöste Flußdiagramm macht diese Denk- und Arbeitsweise recht deutlich.
Eine sehr einfache Aufgabe für den Rechner ist es, nach Eingabe einer natürlichen Zahl n die kleinste Zahl zu finden, die größer als n und durch 31 teilbar ist. Die Struktur des Bearbeitungsprozesses, der Algorithmus, kann so wie in der Abbildung aussehen.
Wer nur ein bißchen Erfahrungen mit BASIC hat, kann diese Struktur leicht in ein Programm übersetzen. Stur geht der Rechner die natürlichen Zahlen durch, wobei er mit der ersten Zahl beginnt, die größer als die eingegebene ist, ihre restlose Teilbarkeit durch 31 überprüft und mit der nächstgrößeren Zahl fortfährt, wenn die vorgegebene Bedingung nicht erfüllt ist.
Prozedurale Sprachen kommen nicht umhin, das zu bearbeitende Problem restlos zu strukturieren und alle Eventualitäten vorauszuplanen. Genau darin liegt aber ihre Schwäche. Wird das Abbruchkriterium für einen bestimmten Arm des Programmflusses mangelhaft definiert, ein Sprungziel falsch bestimmt oder ähnliches, kann das Programm zu falschen Ergebnissen gelangen. In der Regel ist es recht mühsam, diese Fehler (»bugs«) aufzuspüren, denn bei komplexeren Programmen ergeben sich so viele Kreuz- und Querverbindungen zwischen den einzelnen Programmabschnitten, daß strukturelle Abwege und »Trampelpfade« im Programm nur schwer zu entdecken sind.
Speziell für KI-Anwendungen entwickelte Programmiersprachen basieren auf einem grundlegend anderen Konzept. Der Programmierer bestimmt nur die Beziehungen zwischen Objekten und Eigenschaften. Das Finden des Lösungsweges ist Teil der Sprachstruktur und wird vom System in diesem Sinne selbständig geleistet. Der Programmierer ist davon entlastet, den zur Lösung führenden Bearbeitungsprozeß zu strukturieren, er bestimmt vielmehr die logischen Beziehungen innerhalb des Datenmaterials. Programmiersprachen dieser Art werden deshalb deklarativ genannt.
Dem BASIC-Programmierer bleibt es nicht erspart, sein Programm vollständig zu strukturieren, gleichgültig wie verschlungen und verwirrend es auch ausfallen mag. Um ein Expertensystem schreiben zu können, muß ferner ein Weg gefunden werden, den Objekten Eigenschaften (Prädikate) zuzuordnen.



Dieses Flußdiagramm beschreibt den Algorithmus, der die kleinste Zahl findet, die größer als eine eingegebene Zahl »n« und durch 31 teilbar ist.

Um die Schreibgeräte Bleistift, Filzstift, Füller und Kugelschreiber zu erkennen und zu unterscheiden, genügen zwei Eigenschaften: Mine oder Tinte und nachfüllbar oder nicht nachfüllbar. In BASIC würde man die Objekte in Stringvariablen ablegen, zur besseren Bearbeitung in ein String-Array. Die Eigenschaften würden durch numerische Variablen beschrieben, wobei man zwei verschiedene Variablen verwenden könnte. Die eine würde mit 0 oder 1 anzeigen, ob das zugeordnete Objekt mit einer Mine oder mit Tintenflüssigkeit schreibt, die zweite, ob das Schreibgerät nachfüllbar oder nicht nachfüllbar ist. Da jede dieser Alternativen nur ein Bit Informationsgehalt hat, könnten beide Informationen auch zu einem Byte zusammengerechnet werden.

Der BASIC-Experte könnte dann so programmiert sein:

Programm
STIFTE.DEM

```
1 REM STIFTE.DEM
10 FOR j=0 TO 3
20 READ o$(j,0),o$(j,1)
30 NEXT j
100 CLS:MODE 1
110 PRINT"Hat es eine Mine: Ja/Nein":GOS
UB 1000:PRINT:PRINT
120 a=a+k
130 PRINT"Ist es nachfuellbar: Ja/Nein":
GOSUB 1000:PRINT:PRINT
140 a=a+2*k
200 PRINT"Es ist ein ";o$(a,0)
210 PRINT:PRINT:END
1000 PRINT:PRINT:INPUT a$
1010 a$=CHR$(ASC(a$))
1020 IF a$="J" OR a$="j" THEN k=1
1030 IF a$="N" OR a$="n" THEN k=0
1040 RETURN
2000 DATA Filzstift,0,Bleistift,1,Fuelle
r,2,Kugelschreiber,3
```

Programmbeschreibung
STIFTE.DEM

In den **Zeilen 10 bis 30** wird das in der **Datazeile 2000** beschriebene Wissen in die Feldvariable o$ gelesen.

Zeile 100 löscht den Bildschirm und bestimmt den Ausgabemodus. Dann veranlaßt **Zeile 110** die Frage, ob das gesuchte Schreibgerät eine Mine hat. Die Benutzereingabe wird im Unterprogramm ab **Zeile 1000** aufgenommen und umgesetzt. Dabei bekommt die Variable k den Wert 1, wenn die Antwort positiv ist, und 0 bei verneinender Eingabe.

In **Zeile 120** wird der gefundene Wert k an die Variable a übergeben. Dann fragt das Programm, ob das Schreibgerät nachfüllbar ist. Die Antwort wird wieder im Unterprogramm auf die beschriebene Weise verar-

beitet. Der gefundene Wert k wird diesmal jedoch mit 2 multipliziert, um diese Eigenschaft im zweiten Bit der numerischen Variablen a abzulegen.
Danach ist der numerische Wert von a ein Index auf die Variable im Array o$, die dem gesuchten Objekt entspricht. Sein Name ist in o$(a,0) abgelegt. **Zeile 200** gibt dieses Ergebnis aus.
Dieses kleine Beispiel enthält zwar eine Art Wissensbasis in Form der Data-Zeile und ist zu einer bescheidenen Schlußfolgerung in der Lage, aber es ist vollkommen auf diese eine Aufgabe und die vorhandenen Daten beschränkt. Im folgenden Beispiel soll das Programm im Dialog mit dem Benutzer selbständig eine Regel für die Schlußfolgerung entwickeln, also die Komponente Wissenserwerb erhalten.
Das Programm BESTIMM.EXP kann zwischen beliebigen Objekten unterscheiden, die durch eine beliebige Menge von variablen Eigenschaften charakterisiert sind, wenn durch diese vorgegebenen Eigenschaften eine eindeutige Bestimmung möglich ist. Es erfragt zu Beginn die Menge der Merkmale, ihre Bezeichnung und die beiden zu unterscheidenden Objekte.
Der Benutzer denkt sich jetzt eines der beiden Objekte, und während das Programm alle vorgegebenen Merkmale durchgeht, beantwortet er die Programmabfragen, ob das jeweilige Merkmal zutrifft, durch Tastatureingabe. Diese Eingaben werden als numerische Werte 0 oder 1 in einer Feldvariablen gespeichert.
In einer zweiten Feldvariablen entwickelt das Programm seine Schlußfolgerungsregel. Diese zweite Variable ist bei Programmstart leer. Nachdem der Benutzer alle Fragen nach Eigenschaften beantwortet hat, werden die Elemente der ersten Feldvariablen mit den entsprechenden Elementen der zweiten multipliziert und zu einem Wert addiert. Da alle Elemente der zweiten Variablen noch den Wert 0 haben, ist die Summe beim ersten Durchlauf 0.
Das Programm entscheidet sich anhand dieser Summe für eines der beiden Objekte. Ist die Summe größer oder gleich 0, tippt es auf das erste Objekt; ist sie kleiner, auf das zweite. Beim ersten Durchlauf wird also immer das erste Objekt für das gesuchte gehalten.
Trifft dies zu, kann eine neue Objektbestimmung durchgeführt werden. Handelt es sich aber nicht um das gesuchte Objekt, muß die Entscheidungsregel modifiziert werden. Das geschieht, indem für jedes Element der Feldvariablen, in der die Entscheidungsregel gefaßt ist, ein modifizierter Wert ermittelt wird. Dann beginnt ein neuer Programmdurchlauf.

Wenn die Eingaben des Benutzers vollständig und fehlerfrei sind, hat das Programm nach zwei falschen Prognosen eine unfehlbare Entscheidungsregel für die Unterscheidung der beiden Objekte entwickelt. Vorausgesetzt natürlich, die gewählten Merkmale ermöglichen eine Identifikation.

Gemeinsame Merkmale komplizieren die Entscheidungsfindung

Es kann sein, daß die beiden Objekte gemeinsame Merkmale haben. So verfügen sowohl ein Vogel als auch ein Flugzeug über einen Rumpf und Flügel. Allein mit diesen beiden Merkmalen kann also keine Bestimmung vorgenommen werden. Unterscheidungskräftige Eigenschaften treffen auf ein Objekt zu, während sie beim anderen nicht zutreffen.

Beantwortet der Benutzer die Frage nach dem Vorhandensein eines Merkmals als zutreffend, wird die entsprechende Variable mit dem Wert 1 geladen. Die Charakterisierung des Objekts besteht also in der Zuweisung von Nullen (Nein) und Einsen (Ja) zu den einzelnen Elementen der Feldvariablen.

Die Entscheidungsregel entsteht durch Subtraktion bzw. Addition der Werte aus dieser Feldvariablen von bzw. zu den Elementen der Feldvariablen, welche die Entscheidungsregel enthält.

Es soll nun zwischen den Objekten 1 (Vogel) und 2 (Flugzeug) unterschieden werden. Dazu benennen wir sechs Merkmale: Fahrgestell, Federn, Flügel, Krallen, Motor und Rumpf. Die Antworten für Objekt 2 bewirken dann folgende Werte in der Variablen v1(n):

1	Fahrgestell	1
2	Federn	0
3	Flügel	1
4	Krallen	0
5	Motor	1
6	Rumpf	1

Da die Elemente der zweiten Variablen v2(n) noch alle den Wert 0 haben, tippt das Programm auf »Vogel«. Wir verneinen und bewirken damit die Anpassung der Entscheidungsregel, bei der von jedem Wert v2(j) der Wert von v1(j) abgezogen wird:

1	Fahrgestell	0 — 1 = —1
2	Federn	0 — 0 = 0
3	Flügel	0 — 1 = —1
4	Krallen	0 — 0 = 0
5	Motor	0 — 1 = —1
6	Rumpf	0 — 1 = —1

Um das Spiel abzukürzen, beantworten wir beim zweiten Durchlauf die Merkmalfragen in Hinblick auf Vogel:

1	Fahrgestell	0
2	Federn	1
3	Flügel	1
4	Krallen	1
5	Motor	0
6	Rumpf	1

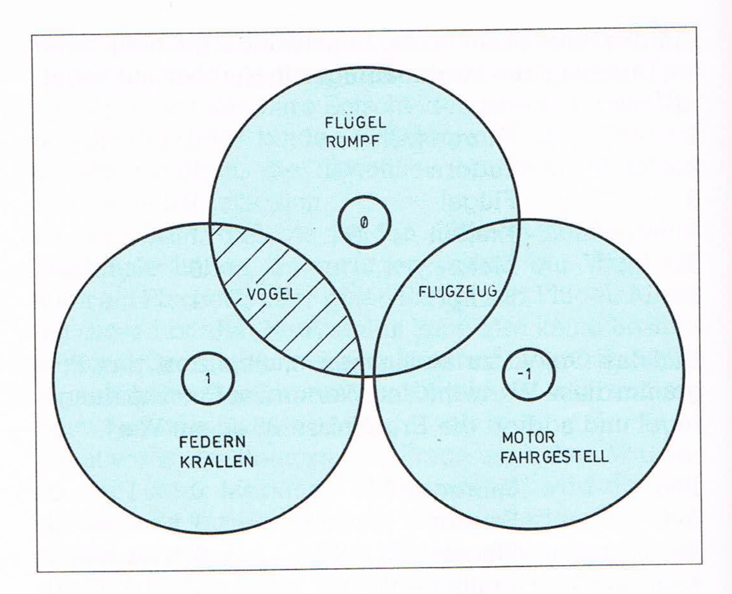
Um das Objekt zu bestimmen, multipliziert das Programm diese Werte mit den Werten der Entscheidungsregel und addiert die Ergebnisse zu einem Wert:

1	Fahrgestell	0 * —1 = 0
2	Federn	1 * 0 = 0
3	Flügel	1 * —1 = —1
4	Krallen	1 * 0 = 0
5	Motor	0 * —1 = 0
6	Rumpf	1 * —1 = —1

Die Summe dieser Werte ist negativ, deshalb tippt das Programm auf Objekt 2 (Flugzeug). Wir verneinen, und die Entscheidungsregel wird erneut modifiziert, in diesem Fall aber durch Addition von v2(j) und v1(j):

1	Fahrgestell	—1 + 0 = —1
2	Federn	0 + 1 = 1
3	Flügel	—1 + 1 = 0
4	Krallen	0 + 1 = 1
5	Motor	—1 + 0 = —1
6	Rumpf	—1 + 1 = 0

Jetzt ist die Entscheidungsregel komplett, Vögel und Flugzeuge werden eindeutig klassifiziert, die Regel also auch nicht mehr verändert.
Bei genauerer Betrachtung der Elemente von v2(n) zeigt sich, daß die Merkmale Flügel und Rumpf, die zur Klassifizierung nicht hinreichen, den Wert 0 angenommen haben. Merkmale, die nur für einen Vogel zutreffen, kennzeichnet eine 1. Merkmale, die nur für ein Flugzeug zutreffen, haben den Wert —1.
Die Abbildung auf S. 132 zeigt die Merkmale auf drei Mengen verteilt, die sich darin unterscheiden, ob der zugehörige Wert v2(j) 1, 0 oder —1 ist. Die Menge 0 enthält {Flügel, Rumpf, Vogel, Flugzeug}. Durch Überschneidung mit der Menge 1 {Federn, Krallen, Vogel} entsteht die Überschneidungsmenge {Vogel}.



Die drei Mengen enthalten die verschiedenen Eigenschaften, die zur Klassifikation zur Verfügung stehen. Die Überschneidungsmengen entsprechen den zu bestimmenden Gegenständen.

Nach diesem harten Stück Arbeit bedeutet es nun keine besondere Schwierigkeit mehr, den Algorithmus in BASIC zu formulieren:

Programm BESTIMM.EXP

```
1 REM BESTIMM.EXP
10 CLS
20 INPUT"Wie viele Merkmale:";n
30 DIM v1(n),v2(n),v$(n)
40 FOR j=1 TO n:v1(j)=0:v2(j)=0:NEXT j
50 PRINT:PRINT"Bitte benennen Sie die Me
rkmale"
60 FOR j=1 TO n:PRINT:PRINT"Merkmal";j;:
INPUT v$(j):NEXT j
70 PRINT:PRINT"Benennen Sie die zu besti
mmenden Objekte"
80 PRINT:INPUT"Objekt 1:";o1$
90 PRINT:INPUT"Objekt 2:";o2$
100 CLS
110 PRINT"Bitte beantworten Sie folgene
Fragen"
120 PRINT"------------------------------
------"
130 FOR j=1 TO n
140 v1(j)=0
150 PRINT:PRINT"Merkmal ";v$(j)
160 PRINT:INPUT"Trifft diese Merkmal zu?
 <J/N>";a$
170 IF a$="J" OR a$="j" THEN v1(j)=1
180 NEXT j
200 e=0
```

Programm
BESTIMM.EXP

```
210 FOR j=1 TO n:e=e+v1(j)*v2(j):NEXT j
220 IF e>=0 THEN PRINT:PRINT"Es ist ein/
e ";o1$:PRINT:INPUT"Ist das richtig? <J/
N>";a$:IF a$="J" OR a$="j" GOTO 100
230 IF e<0 THEN PRINT:PRINT"Es ist ein/e
 ";o2$:PRINT:INPUT"Ist das richtig? <J/N
>";a$:IF a$="J" OR a$="j" GOTO 100
240 IF e>=0 AND (a$="N" OR a$="n") THEN
FOR j=1 TO n:v2(j)=v2(j)-v1(j):NEXT j
250 IF e<0 AND (a$="N" OR a$="n") THEN F
OR j=1 TO n:v2(j)=v2(j)+v1(j):NEXT j
260 GOTO 100
```

Programmbeschreibung
BESTIMM.EXP

Zeile 20 nimmt in der Variablen n die Anzahl der Merkmale auf, und **Zeile 30** dimensioniert entsprechend die Feldvariablen v1, v2 und v$. Die Bezeichnungen der Merkmale werden in v$ abgelegt; v1 nimmt die Merkmale auf, wie sie der Benutzer später antwortend eingibt, und in v2 wird die Entscheidungsregel entwickelt. Die FOR-NEXT-Schleife in **Zeile 40** setzt alle Elemente von v1 und v2 auf 0.
In den **Zeilen 50 bis 90** wird der Benutzer aufgefordert, die Merkmale und die beiden zu unterscheidenden Objekte zu benennen. Dann geht es los.
Nach dem Löschen des Bildschirms (**Zeile 100**) geht das Programm alle Merkmale durch, schreibt ihre Bezeichnung auf den Bildschirm (**Zeile 150**) und nimmt die Benutzerantwort auf (**Zeile 170**).
Ab **Zeile 200** beginnt die Auswertung, deren summarisches Ergebnis in der Variablen e addiert wird, die vorab auf 0 gesetzt wird. **Zeile 210** errechnet aus den Produkten der korrespondierenden Elemente aus v1 und v2 den Wert für e.
Ist e größer oder gleich 0, tippt das Programm in **Zeile 220** auf das Objekt als o1$. Beantwortet der Benutzer diesen Versuch, die Lösung zu erraten, mit »Ja«, beginnt mit dem Befehl GOTO 100 ein neuer Bestimmungsdurchlauf.
Ist e kleiner 0, vermutet das Programm, daß Objekt o2$ der gesuchte Begriff ist. Ist diese Vermutung richtig, wird ebenfalls nach **Zeile 100** verzweigt.
Erfolgt aber die Benutzerantwort »Nein«, gelangt das Programm zur **Zeile 240**, die nur bearbeitet wird, wenn e größer oder gleich 0 ist. Dort werden die Werte der Elemente aus der Feldvariablen v2 durch Substraktion der entsprechenden Elemente aus v1 verändert.
Ist e kleiner 0, werden die Elemente von v2 in **Zeile 250** durch Addition mit den entsprechenden Elementen aus v1 verändert.

Programmbeschreibung BESTIMM.EXP

Zeile 260 beendet den Durchlauf mit einem Rücksprung nach **Zeile 100**. Das Programm hat also die Form einer endlosen Schleife und kann nur durch externen Abbruch (zweimal <ESC> drücken) beendet werden.
Das Fachwissen eines solchen Expertensystems besteht darin, dem Benutzer Merkmale vorzulegen und seine Antworten mit Werten zu belegen. Diese Werte werden mit Hilfe der Entscheidungsregel in ein Ergebnis verwandelt.
Probieren Sie das Programm ein wenig aus. Geben Sie verschiedene Objekte mit wechselnden Merkmalen ein. Stellen Sie fest, wie das Programm bei falschen oder lückenhaften Eingaben reagiert, und versuchen Sie, die Ergebnisse anhand des Listings nachzuvollziehen.

Ein kleiner Psychotest

Das nun folgende Programm ist ein fertiges kleines Expertensystem mit einem psychologischen Test. Es legt dem Benutzer eine Reihe von Fragen und jeweils drei mögliche Antworten vor. Die Summe der Antworten führt abschließend zu einer charakterlichen Beurteilung.
Bevor wir den Aufbau dieses Programms genauer unter die Lupe nehmen, sollten Sie es ganz unvoreingenommen ausprobieren:

Programm PSYCHO.EXP

```
1 REM PSYCHO.EXP
10 GOSUB 40010
20 GOSUB 30010
1000 REM > Frage 1 <
1010 LOCATE 4,3:PRINT"Denken Sie beim Ba
den am"
1020 PRINT"     Meer manchmal an Haie,"
1030 PRINT"  weil Sie durch Filme oder"
1040 PRINT" BÜcher Über diese Raubfische
"
1050 PRINT"       beeinflußt wurden?"
1060 LOCATE #2,2,2:PRINT#2,"<a> Manchmal
 habe ich"
1070 PRINT#2,"     ein komisches GefÜhl"
1080 PRINT#2," <b> Nein, dieser Gedanke"
1090 PRINT#2,"     beschaftigt mich nie"
1100 PRINT#2," <c> Ja, ziemlich oft"
1110 PRINT#2,"     beschleicht mich Angs
t"
1120 i$=INKEY$
1130 IF I$="a" OR I$="A" THEN z=z+3:GOTO
 1170
1140 IF I$="b" OR I$="B" THEN z=z+4:GOTO
 1170
1150 IF I$="c" OR I$="C" THEN z=z+1:GOTO
 1170
1160 GOTO 1120
```

Programm
PSYCHO.EXP

```
1170 CLS:CLS#2
2000 REM > Frage 2 <
2010 LOCATE 8,4:PRINT"Legen Sie Wert"
2020 PRINT:PRINT"      auf Umgangsformen
?"
2030 LOCATE #2,2,2:PRINT#2,"<a> Nein, ke
inen besonderen"
2040 PRINT#2:PRINT#2," <b> Nur bei wicht
igen"
2050 PRINT#2,"                      Anläs
sen"
2060 PRINT#2," <c> Ja, immer"
2070 i$=INKEY$
2080 IF I$="a" OR I$="A" THEN z=z+1:GOTO
 2120
2090 IF I$="b" OR I$="B" THEN z=z+4:GOTO
 2120
2100 IF I$="c" OR I$="C" THEN z=z+3:GOTO
 2120
2110 GOTO 2070
2120 CLS:CLS#2
3000 REM > Frage 3 <
3010 LOCATE 3,3:PRINT"Worauf kommt es vo
r allem"
3020 PRINT:PRINT"   an, wenn man berufli
chen"
3030 PRINT:PRINT"      Erfolg haben will
?"
3040 LOCATE #2,2,2:PRINT#2,"<a> Auf die
persönliche"
3050 PRINT#2,"                      Begab
ung"
3060 PRINT#2," <b> Auf den Fleiß"
3070 PRINT#2:PRINT#2," <c> Auf das Glück
"
3080 i$=INKEY$
3090 IF I$="a" OR I$="A" THEN z=z+3:GOTO
 3130
3100 IF I$="b" OR I$="B" THEN z=z+5:GOTO
 3130
3110 IF I$="c" OR I$="C" THEN z=z+1:GOTO
 3130
3120 GOTO 3080
3130 CLS:CLS#2
4000 REM > Frage 4 <
4010 LOCATE 3,2:PRINT"Wie finden Sie ein
e fremd-"
4020 PRINT:PRINT"     ländische Maske mi
t"
4030 PRINT:PRINT"  stechenden Augen, Fla
mmen-"
4040 PRINT:PRINT"  haaren und knorriger
Nase?"
4050 LOCATE #2,2,2:PRINT#2,"<a> Interess
ant"
4060 PRINT#2:PRINT#2," <b> Häßlich"
4070 PRINT#2:PRINT#2," <c> Bedrohlich"
4080 i$=INKEY$
4090 IF I$="a" OR I$="A" THEN z=z+5:GOTO
 4130
```

Programm
PSYCHO.EXP

```
4100 IF I$="b" OR I$="B" THEN z=z+3:GOTO
 4130
4110 IF I$="c" OR I$="C" THEN z=z+2:GOTO
 4130
4120 GOTO 4080
4130 CLS:CLS#2
5000 REM > Frage 5 <
5010 LOCATE 4,3:PRINT"Würden Sie es als
störend"
5020 PRINT"     empfinden, von einem"
5030 PRINT" verwegen gekleideten jungen"
5040 PRINT"      Mann auf der Straße"
5050 PRINT"   angesprochen zu werden?"
5060 LOCATE #2,2,2:PRINT#2,"<a> Ja, ich
glaube schon"
5070 PRINT#2:PRINT#2," <b> Das kann ich
so nicht"
5080 PRINT#2,"                    beurtei
len"
5090 PRINT#2," <c> Nein, gewiß nicht"
5100 i$=INKEY$
5110 IF I$="a" OR I$="A" THEN z=z+2:GOTO
 5150
5120 IF I$="b" OR I$="B" THEN z=z+3:GOTO
 5150
5130 IF I$="c" OR I$="C" THEN z=z+5:GOTO
 5150
5140 GOTO 5100
5150 CLS:CLS#2
6000 REM > Frage 6 <
6010 LOCATE 2,3:PRINT"Könnte es Ihnen Sp
aß machen,"
6020 PRINT:PRINT"   einen Dschungelpfad
auf"
6030 PRINT:PRINT"    Hawaii zu erforsche
n?"
6040 LOCATE #2,2,2:PRINT#2,"<a> Ja"
6050 PRINT#2:PRINT#2," <b> Nur in erfahr
ener"
6060 PRINT#2,"                    Begleit
ung"
6070 PRINT#2," <c> Nein"
6080 i$=INKEY$
6090 IF I$="a" OR I$="A" THEN z=z+5:GOTO
 6130
6100 IF I$="b" OR I$="B" THEN z=z+3:GOTO
 6130
6110 IF I$="c" OR I$="C" THEN z=z+1:GOTO
 6130
6120 GOTO 6080
6130 CLS:CLS#2
10000 REM > Auswertung <
10010 MODE 1:BORDER 0:INK 0,0:PAPER 0:CL
S
10020 WINDOW 6,35,5,20:WINDOW #1,1,40,25
,25
10030 INK 1,2:PEN #1,1
10040 PRINT#1,"< < Weiter, beliebige Tas
te drücken > >";
10050 IF z<13 THEN GOSUB 11010
```

Programm
PSYCHO.EXP

```
10060 IF z>12 AND z<23 THEN GOSUB 12010
10070 IF z>22 THEN GOSUB 13010
10080 MODE 0:CLS
10090 BORDER 2:INK 0,2:INK 1,1:PAPER 0:C
LS
10100 PEN 1:LOCATE 9,5:PRINT"ENDE":GOTO
10100
11000 REM > Ergebnis 1 <
11010 INK 2,15:INK 3,3:PAPER 2:PEN 3:CLS
11020 LOCATE 2,2
11030 PRINT"Sie sind ein Mensch, der"
11040 PRINT:PRINT" nach Möglichkeit alle
n"
11050 PRINT:PRINT" Auseinandersetzungen"
11060 PRINT:PRINT" aus dem Weg geht und"
11070 GOSUB 20010
11100 PRINT"die Devise"
11110 PRINT:PRINT" 'Der Klügere gibt nac
h'"
11120 PRINT:PRINT" zur Richtschnur seine
s"
11130 PRINT:PRINT" Verhaltens gemacht ha
t."
11140 GOSUB 20010
11200 PRINT"Damit liegen Sie zwar oft"
11210 PRINT:PRINT" richtig - aber eben"
11220 PRINT:PRINT" doch nicht immer, den
n"
11230 GOSUB 20010
11300 PRINT"es gibt nun mal gelegentlich
"
11310 PRINT:PRINT" Situationen, in denen
 man"
11320 PRINT:PRINT" sich nur behaupten ka
nn,"
11330 PRINT:PRINT" wenn man sich wehrt."
11340 GOSUB 20010
11400 PRINT"Aber das fällt Ihnen"
11410 PRINT:PRINT" meist recht schwer."
11420 PRINT:PRINT" Außerdem scheuen Sie"
11430 PRINT:PRINT" davor zurück, energis
ch"
11440 PRINT:PRINT" aufzutreten."
11450 PRINT:PRINT" Das widerspricht Ihre
m"
11460 PRINT:PRINT" Naturell."
11470 GOSUB 20010
11500 RETURN
12000 REM > Ergebnis 2 <
12010 INK 2,4:INK 3,11:PAPER 2:PEN 3:CLS
12020 LOCATE 2,2
12030 PRINT"Sie können sich zwar"
12040 PRINT:PRINT" zur Wehr setzen -"
12050 PRINT:PRINT" aber Sie tun das nur,
"
12060 PRINT:PRINT" wenn es unbedingt sei
n muß."
12070 PRINT:PRINT" Bei nicht so bedeuten
den
12080 PRINT:PRINT" Anlässen"
```

Programm
PSYCHO.EXP

```
12090 GOSUB 20010
12100 PRINT"geben Sie lieber etwas nach,
"
12110 PRINT:PRINT" als durch sofortigen"
12120 PRINT:PRINT" Widerstand vielleicht
"
12130 PRINT:PRINT" mehr kaputtzumachen,
als"
12140 PRINT:PRINT" die ganze Angelegenhe
it"
12150 PRINT:PRINT" wert ist. Nur wenn"
12160 GOSUB 20010
12200 PRINT"diese Grenze überschritten"
12210 PRINT:PRINT" wird, wenn es um wirk
lich"
12220 PRINT:PRINT" Wesentliches geht,"
12230 PRINT:PRINT" setzen Sie sich"
12240 PRINT:PRINT" nachhaltig zur Wehr."
12250 GOSUB 20010
12300 PRINT"Es bereitet Ihnen aber"
12310 PRINT:PRINT" kein Vergnügen, denn
Sie"
12320 PRINT:PRINT" sind ein Mensch, für
den"
12330 PRINT:PRINT" Harmonie sehr wichtig
 ist."
12340 GOSUB 20010
12400 PRINT"Diese Lebenseinstellung"
12410 PRINT:PRINT" ist im Prinzip"
12420 PRINT:PRINT" auch richtig. Nur sol
lten"
12430 PRINT:PRINT" Sie auch überlegen:"
12440 GOSUB 20010
12500 PRINT"Ein klärendes Gespräch,"
12510 PRINT:PRINT" auch wenn dabei die"
12520 PRINT:PRINT" Fetzen fliegen,"
12530 PRINT:PRINT" hat oftmals eine"
12540 PRINT:PRINT" reinigende Wirkung."
12550 GOSUB 20010
12600 RETURN
13000 REM > Ergebnis 3 <
13010 INK 2,24:INK 3,6:PAPER 2:PEN 3:CLS
13020 LOCATE 2,2
13030 PRINT"Sie haben keine"
13040 PRINT:PRINT" Schwierigkeiten, wenn
"
13050 PRINT:PRINT" es gilt, sich zur Weh
r"
13060 PRINT:PRINT" zu setzen. Denn"
13070 GOSUB 20010
13100 PRINT"Sie stehen auf dem"
13110 PRINT:PRINT" Standpunkt, daß man a
uch"
13120 PRINT:PRINT" bei Nebensächlichkeit
en"
13130 PRINT:PRINT" möglichst wenig nachg
eben"
13140 PRINT:PRINT" und"
13150 GOSUB 20010
13200 PRINT"von vorneherein seinen"
```

Programm
PSYCHO.EXP

```
13210 PRINT:PRINT" Verteidigungswillen"
13220 PRINT:PRINT" klarmachen sollte."
13230 GOSUB 20010
13300 PRINT"Nur so kann"
13310 PRINT:PRINT" Ihrer Meinung nach"
13320 PRINT:PRINT" Ihre Position klar un
d"
13330 PRINT:PRINT" unangefochten bleiben
."
13340 GOSUB 20010
13400 PRINT"Dabei übersehen Sie"
13410 PRINT:PRINT" allerdings manchmal,"
13420 PRINT:PRINT" daß das Leben ohne"
13430 PRINT:PRINT" Kompromisse"
13440 PRINT:PRINT" kaum möglich ist - un
d"
13450 PRINT:PRINT" Kompromisse bestehen
nun mal"
13460 PRINT:PRINT" aus Nachgeben."
13470 GOSUB 20010
13500 PRINT"Auch daran sollten Sie"
13510 PRINT:PRINT" gelegentlich denken."
13520 PRINT:PRINT" Denn es kann sein, da
ß"
13530 PRINT:PRINT" sich Ihre Mitmenschen
"
13540 PRINT:PRINT" durch Ihr Verhalten"
13550 PRINT:PRINT" vor den Kopf gestoßen
"
13560 PRINT:PRINT" fühlen."
13570 GOSUB 20010
13600 PRINT"Sie konnen dann leicht"
13610 PRINT:PRINT" in den Ruf kommen,"
13620 PRINT:PRINT" egoistisch zu sein."
13630 PRINT:PRINT" Und es ist immer schw
er,"
13640 PRINT:PRINT" ein einmal gefälltes"
13650 PRINT:PRINT" Vorurteil"
13660 PRINT:PRINT" zu korrigieren."
13670 GOSUB 20010
13700 RETURN
20000 REM > UP Tastaturabfrage <
20010 IF INKEY$="" THEN GOTO 20010
20020 CLS:LOCATE 2,2
20030 RETURN
30000 REM > Bildschirmgestaltung <
30010 BORDER 6:INK 0,6:INK 1,0:INK 2,21
30020 MODE 0:PAPER 0
30030 LOCATE 7,4:PEN 1:PRINT"PSYCHO"
30040 LOCATE 8,6:PEN 2:PRINT"TEST"
30050 FOR j=0 TO 5:NEXT j
30060 INK 0,0:INK 1,22:INK 2,21:INK 3,6
30070 BORDER 0:MODE 1
30080 WINDOW #1,1,40,1,7:PEN #1,1
30090 FOR j=0 TO 39:PRINT #1,CHR$(131);:
NEXT j
30100 PRINT #1,"  Wählen Sie aus den dre
i vorgegebenen"
30110 PRINT #1," Antworten die für Sie p
assende aus und"
```

Programm
PSYCHO.EXP

```
30120 PRINT #1,"geben sie durch Drücken
der entsprechen-";
30130 PRINT #1,"          Buchstabentaste
<_> ein."
30140 FOR j=0 TO 39:PRINT #1,CHR$(140);:
NEXT j
30150 WINDOW #0,6,35,9,17:PAPER 2:CLS:PE
N 3
30160 WINDOW #2,6,35,19,25:PAPER #2,3:CL
S #2:PEN #2,2
30170 RETURN
40000 REM > Umlaute <
40010 SYMBOL AFTER 91
40020 SYMBOL 91,66,24,60,102,126,102,102
,0
40030 SYMBOL 92,130,56,108,198,198,108,5
6,0
40040 SYMBOL 93,66,0,102,102,102,102,60,
0
40050 SYMBOL 123,68,0,120,12,124,204,118
,0
40060 SYMBOL 124,36,0,60,102,102,102,60,
0
40070 SYMBOL 125,36,0,102,102,102,102,62
,0
40080 SYMBOL 126,60,102,124,102,102,102,
108,96
```

Programmbeschreibung
PSYCHO.EXP

Im Unterprogramm ab **Zeile 40010** werden die deutschen Sonderzeichen definiert. Das Unterprogramm ab **Programmzeile 30010** organisiert die Bildschirmgestaltung.

Die **Zeilen 1010 bis 1050** bringen die erste Frage auf den Bildschirm, die **Zeilen 1060 bis 1110** die drei Antwortmöglichkeiten. **Zeile 1120** übernimmt die Benutzereingabe. Die **Zeilen 1130 bis 1150** setzen die Eingabe in einen numerischen Wert um, der in der Variablen z, auch bei den folgenden Fragen, fortlaufend erfaßt wird. **Zeile 1160** führt zur Tastaturabfrage zurück, falls der Benutzer keine zulässige Eingabe gemacht hat. In **Zeile 1170** wird der Bildschirm gelöscht.

Auf die gleiche Weise werden dem Benutzer in den **Zeilen 2000 bis 6130** fünf weitere Fragen vorgelegt. Dann folgt die Auswertung.

Das Auswertungsverfahren ist in diesem Fall denkbar einfach. Das Programm hält drei Urteile bereit. Der in der Variablen z angesammelte Wert bestimmt, welches Urteil ausgegeben wird. Diese Auswahl erfolgt in den **Zeilen 10050 bis 10070**.

Wenn die Punktesumme z kleiner als 13 ist, wird Ergebnis 1 auf dem Bildschirm ausgegeben. Die **Zeilen 11000 bis 11500** besorgen diese Arbeit. Der recht lange Text

wird in einzelnen Abschnitten auf den Bildschirm gebracht. Der Benutzer blättert durch Betätigen einer beliebigen Taste weiter. Zu diesem Zweck wird wiederholt das Unterprogramm ab **Zeile 20010** aufgerufen.
Die Ergebnisse 2 und 3 werden gegebenenfalls von den **Programmzeilen 12000 bis 13700** ausgegeben. Nach diesem »psychologischen Urteil« läßt das Programm den Benutzer mit einem spröden ENDE allein.

Intelligente Dateien

Alle bisher in diesem Buch vorgestellten Programme dienten lediglich der Demonstration und waren nichts als amüsante Spielereien. Aber zum Abschluß soll die ganze künstlich-intelligente Anstrengung auch noch eine nützliche Frucht hervorbringen: ein Datenverarbeitungsprogramm mit einer intelligenten Struktur.
Bei üblichen Datenverwaltungsprogrammen werden verschiedene Einzeldaten (Datenfelder) zu einem Datensatz zusammengefaßt. Eine Adresse ist zum Beispiel ein Datensatz, der aus den Feldern Name, Straße und Ort bestehen kann. Jeder Datensatz bekommt vom Programm intern eine Nummer zugewiesen, womit eine Zugriffsmöglichkeit erschlossen wird.
Untereinander stehen die einzelnen Datensätze in keiner Verbindung. Das Programm kann zwar den Datenbestand nach einem bestimmten Merkmal durchsuchen und die entsprechenden Datensätze ausgeben, aber konventionelle Datenbanken arbeiten immer nach dem Lexikonprinzip: Nur wenn man das Stichwort kennt, findet man die gesuchten Informationen. Wenn in Ihrem Computer ein Datenbestand von 1000 Adressen gespeichert ist und Sie suchen die Anschrift einer Person, deren Namen Sie nicht mehr genau wissen, hilft Ihnen das System auch nicht viel weiter.
Das menschliche Gehirn arbeitet vollkommen anders. Seine Leistungsfähigkeit basiert im wesentlichen nicht auf reiner Wissensanhäufung, sondern auf der mächtigen Zahl assoziativer Verflechtungen zwischen den Informationsteilen.

Assoziationen: Verknüpfungen zwischen Informationen sind der entscheidende Vorsprung des menschlichen Gehirns

Die Assoziationen erfolgen zum Teil sachbezogen. Zu einem Musikinstrument fällt mir ein zweites ein. Teilweise sind sie aber auch durch individuelle Erfahrungen geprägt. Wenn ich chinesisch essen gehe, denke ich oft an meine Hochzeit. Die Feier fand in einem chinesischen Restaurant statt.

Intelligenz ist also nicht die Summe aller im Gehirn gespeicherten Daten. Die geheimnisvollen und teilweise irrationalen Verflechtungen der Gedanken und das (zufällige?) Zusammentreffen beliebiger Gedanken, das einen neuen Impuls auslöst (Intuition, Kreativität), sind für die Entwicklung unserer Kultur von besonderer Bedeutung.

DENKNETZ:
Ein elektronisches
Kurzzeitgedächtnis

DENKNETZ.EXP ist ein elektronisches »Kurzzeitgedächtnis« mit ausgeprägter Lern- und Verflechtungsfähigkeit. In der vorliegenden Form kann es 400 Begriffe aufnehmen und jeden dieser 400 Begriffe mit zehn anderen verknüpfen. So entstehen bis zu 4000 Assoziationen.

Auf dem Bildschirm wird eine Assoziationskette von zehn Begriffen gezeigt. Jeder dem Programm bekannte Begriff kann zum Wurzelbegriff gemacht werden. Er steht auf dem Bildschirm ganz oben. Darunter wird einer der Begriffe gezeigt, mit dem der Wurzelbegriff assoziiert ist. Durch Betätigen der Cursortasten »rechts« und »links« kann durch die maximal zehn assoziierten Begriffe geblättert werden.

Unter diesem Begriff wird wieder ein Begriff gezeigt, der mit dem darüberstehenden verbunden ist. Auch auf dieser Ebene kann geblättert werden. Und so geht es bis hinunter zur neunten Ebene.

Mit den Cursortasten »auf« und »ab« wird ein Merker auf dem Bildschirm bewegt, der auf den Wurzelbegriff oder die eine Assoziationsebene zeigt. Wenn mit den Cursortasten geblättert wird, wechselt zwar nur der Begriff auf dieser aktuellen Ebene; da der neu gezeigte Begriff aber wahrscheinlich andere Assoziationsverknüpfungen hat, ändert sich sofort auch die Anzeige auf allen darunterliegenden Ebenen.

Die Eingabe von Daten ist einfach. Zuerst wird der Merker auf dem Bildschirm in die gewünschte Assoziationsebene gebracht, dann solange geblättert, bis das entsprechende Bildschirmfeld leer ist, die Funktionstaste <f1> gedrückt und schließlich der Text eingetippt. Erfolgt die Eingabe auf der Wurzelbegriffsebene, wird keine Assoziation hergestellt. Auf jeder anderen Ebene ist der eingegebene Begriff automatisch mit dem direkt darüberstehenden verbunden.

Bei fehlerhaften Eingaben oder wenn kein leeres Feld mehr zu finden ist, weil alle zehn Assoziationen belegt sind, kann durch Betätigen von <f2> gelöscht werden. Wird ein Begriff nur auf einer Assoziationsebene ge-

löscht, bleibt er erhalten, und nur die Verknüpfung wird gestrichen. Der Begriff kann dann immer noch als Wurzelbegriff aufgerufen werden oder in anderen assoziativen Verknüpfungen vorkommen. Wird ein Begriff auf der Wurzelebene gelöscht, ist er damit vollkommen eliminiert. Alle Verknüpfungen zu diesem Begriff werden gleichzeitig gelöscht.
Wenn ein Begriff noch keine Assoziationen gebildet hat, erscheint auf allen unter ihm liegenden Ebenen ein Gedankenstrich (—).
Der gesamte Inhalt einer DENKNETZ-Datei kann durch Drücken der Funktionstaste <f7> auf dem Drucker ausgegeben werden. Neben der laufenden Nummer und dem Textinhalt des Begriffs werden die zehn Referenznummern der Assoziationen aufgelistet.
Dabei kann es vorkommen, daß unter einer laufenden Nummer kein Textinhalt erscheint. Das geschieht immer dann, wenn zuvor ein Begriff gelöscht und der freie Platz durch Neueingaben noch nicht wieder aufgefüllt wurde. Das Programm führt Buch über die durch Löschen frei gewordenen Plätze in der Datei und füllt sie bei Neueingaben vorrangig auf.
Durch Betätigen von <f9> wird das Programm beendet. Der aktuelle Inhalt der Datei muß dann auf Diskette gespeichert werden, weil er sonst mit dem Ausschalten des Computers oder Laden eines anderen Programms verlorengehen würde. Die Datendatei kann beliebig benannt werden. Dazu stehen die üblichen acht Anschläge für einen Dateinamen zur Verfügung. Das Programm hängt automatisch den Extender DND an. So ist es möglich, mit DENKNETZ verschiedene Dateien aufzubauen und zu bearbeiten. Bei Programmstart kann sowohl eine vorhandene Datei geladen als auch eine neue eröffnet werden.
Auf der Diskette zu diesem Buch ist die Datei MOZART.DND gespeichert. Sie enthält einen Ausschnitt aus dem Stammbaum von Wolfgang Amadeus Mozart über vier Generationen. Hinter den Namen der Personen sind Ziffern von 1 bis 4 eingegeben, mit denen auf die jeweilige Generationsebene verwiesen wird. Wolfgang Amadeus hat die Generationsziffer 3. Personen mit gleicher Kennziffer gehören folglich zur Geschwistergeneration, Personen mit niedrigerer Kennziffer zu den Eltern bzw. Großeltern, Personen mit Kennziffer 4 zur Kindergeneration. Den Ausdruck der Datei MOZART.DND zeigt die Abbildung.

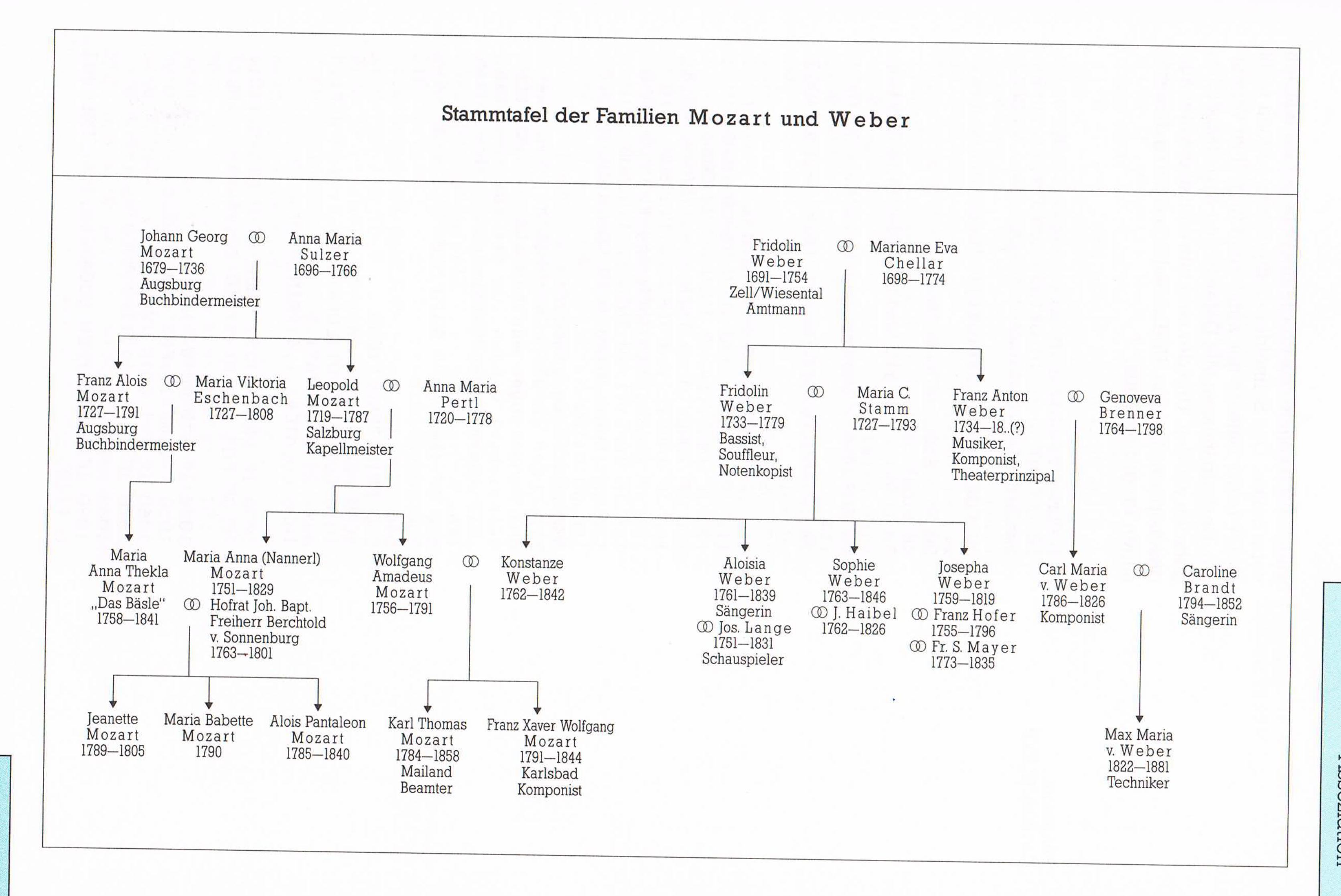
```
Bestand der Datei MOZART.DND

Begriffe mit Referenznummern
------------------------------------------------------------------------

 0: -....................   0   0   0   0   0   0   0   0   0   0
 1: Johann Georg Mozart 1   2   3   4   0   0   0   0   0   0   0
 2: Anna Maria Sulzer 1..   1   3   4   0   0   0   0   0   0   0
 3: Leopold Mozart 2.....   1   2   4   7   8   9   0   0   0   0
 4: Franz Alois Mozart 2.   1   2   3   5   6   0   0   0   0   0
 5: Maria V.Eschenbach 2.   4   6   0   0   0   0   0   0   0   0
 6: Maria A.Th.Mozart 3..   4   5   0   0   0   0   0   0   0   0
 7: Anna Maria Pertl 2...   3   8   9   0   0   0   0   0   0   0
 8: Maria Anna Mozart 3..   3   7   9  10  11  12  13   0   0   0
 9: W.Amadeus Mozart 3...   3   7   8  15  16  17   0   0   0   0
10: J.B.v.Sonnenburg 3...   8  11  12  13   0   0   0   0   0   0
11: Alois P.Mozart 4.....  10   8  12  13   0   0   0   0   0   0
12: Jeanette Mozart 4....  10   8  11  13   0   0   0   0   0   0
13: Maria B.Mozart 4.....  10   8  11  12   0   0   0   0   0   0
14: .....................   0   0   0   0   0   0   0   0   0   0
15: Konstanze Weber 3....  19  24  25  26  27   9  16  17   0   0
16: Karl Thomas Mozart 4.   9  15  17   0   0   0   0   0   0   0
17: Franz X.W.Mozart 4...   9  15  16   0   0   0   0   0   0   0
18: Fridolin Weber 1.....  19  19  20  21   0   0   0   0   0   0
19: Fridolin Weber II 2..  18  21  20  24  25  26  15  27   0   0
20: Franz Anton Weber 2..  18  21  19  22  23   0   0   0   0   0
21: Marianne E.Chellar 1.  18  19  20   0   0   0   0   0   0   0
22: Genoveva Brenner 2...  20  23   0   0   0   0   0   0   0   0
23: Carl Maria v.Weber 3.  20  22  28  29   0   0   0   0   0   0
24: Maria C.Stamm 2......  19  25  26  15  27   0   0   0   0   0
25: Josepha Weber 3......  19  24  26  15  27  30  31   0   0   0
26: Aloisa Weber 3.......  24  25  15  27  32   0   0   0   0   0
27: Sophie Weber 3.......  19  24  25  26  15  33   0   0   0   0
28: Caroline Brandt 3....  23  29   0   0   0   0   0   0   0   0
29: Max Maria v.Weber 4..  23  28   0   0   0   0   0   0   0   0
30: Franz Hofer 3........  25   0   0   0   0   0   0   0   0   0
31: Fr.S.Mayer 3.........  25   0   0   0   0   0   0   0   0   0
32: Jos.Lange 3..........  26   0   0   0   0   0   0   0   0   0
33: J.Haibel 3...........  27   0   0   0   0   0   0   0   0   0
34: .....................   0   0   0   0   0   0   0   0   0   0
```

Wenn Sie nicht über die Diskette zum Buch verfügen, müssen Sie die Daten selbst eintippen. Eine recht umfangreiche Arbeit, da jede verwandtschaftliche Beziehung zwischen zwei Personen eingegeben werden muß, um alle Stammbaum-Assoziationen zu erzeugen. Den Stammbaum, aus dem die Daten für MOZART.DND ermittelt wurden, zeigt die Abbildung.

Eine Schwäche sei noch erwähnt, die aber nicht typisch für dieses Programm, sondern für Datenverarbeitung überhaupt ist. Begriffe werden nur als gleich behandelt, also auch Assoziationen nur richtig hergestellt, wenn sie aus einer identischen Folge von Zeichen bestehen. Das Programm behandelt also »Wolfgang Amadeus Mozart« gesondert von »Wolfgang A. Mozart«, das heißt, Sie dürfen sich bei der Eingabe keinen Tippfehler erlauben.


Stammtafel der Familien Mozart und Weber
Johann Georg Mozart 1679–1736 Augsburg Buchbindermeister
⚭
Anna Maria Sulzer 1696–1766
Fridolin Weber 1691–1754 Zell/Wiesental Amtmann
⚭
Marianne Eva Chellar 1698–1774
Franz Alois Mozart 1727–1791 Augsburg Buchbindermeister
⚭
Maria Viktoria Eschenbach 1727–1808
Leopold Mozart 1719–1787 Salzburg Kapellmeister
⚭
Anna Maria Pertl 1720–1778
Fridolin Weber 1733–1779 Bassist, Souffleur, Notenkopist
⚭
Maria C. Stamm 1727–1793
Franz Anton Weber 1734–18..(?) Musiker, Komponist, Theaterprinzipal
⚭
Genoveva Brenner 1764–1798
Maria Anna Thekla Mozart „Das Bäsle" 1758–1841
Maria Anna (Nannerl) Mozart 1751–1829
⚭ Hofrat Joh. Bapt. Freiherr Berchtold v. Sonnenburg 1763–1801
Wolfgang Amadeus Mozart 1756–1791
⚭
Konstanze Weber 1762–1842
Aloisia Weber 1761–1839 Sängerin
⚭ Jos. Lange 1751–1831 Schauspieler
Sophie Weber 1763–1846
⚭ J. Haibel 1762–1826
Josepha Weber 1759–1819
⚭ Franz Hofer 1755–1796
⚭ Fr. S. Mayer 1773–1835
Carl Maria v. Weber 1786–1826 Komponist
⚭
Caroline Brandt 1794–1852 Sängerin
Jeanette Mozart 1789–1805
Maria Babette Mozart 1790
Alois Pantaleon Mozart 1785–1840
Karl Thomas Mozart 1784–1858 Mailand Beamter
Franz Xaver Wolfgang Mozart 1791–1844 Karlsbad Komponist
Max Maria v. Weber 1822–1881 Techniker

Theoretisch könnte das Programm beliebig viele Daten verarbeiten. Der Schneider-Computer ist jedoch nicht in der Lage, Dateien mit wahlfreiem Zugriff zu verwalten. Deshalb müssen alle Daten in den Arbeitsspeicher geladen werden. Und der setzt nun einmal die hier ausgeschöpften Grenzen. Und so sieht das Listing dieses listigen Programms aus:

Programm DENKNETZ.EXP

```
1 REM DENKNETZ.EXP
10 DEFINT a-z:DIM w$(399),r(399,9),l(99)
:x=20:ex$=".DND":w$(0)="-..............
....."
20 FOR j=1 TO 399:w$(j)="..............
......":NEXT j
30 PEN 2:CLS:GOSUB 5010
40 GOSUB 6010
50 CLS#1:PRINT#1,"Wollen Sie eine besteh
ende Datei"
60 INPUT #1,"weiter bearbeiten? - <J/N>"
;t$
70 IF ASC(t$)=74 OR ASC(t$)=106 THEN GOS
UB 9010
80 GOSUB 7010
100 IF INKEY(3)=0 GOTO 510
110 IF INKEY(13)=0 THEN GOSUB 1010
120 IF INKEY(14)=0 THEN GOSUB 2010
130 IF INKEY(0)=0 THEN ee=-1:GOSUB 4010
140 IF INKEY(2)=0 THEN ee=1:GOSUB 4010
150 IF INKEY(8)=0 THEN pp=-1:GOSUB 3010
160 IF INKEY(1)=0 THEN pp=1:GOSUB 3010
170 IF INKEY(10)=0 THEN GOSUB 10010
180 GOTO 100
500 REM >Programmende<
510 CLEAR INPUT:CLS#1:PRINT#1,"Eingaben
und Aenderungen muessen auf     Diskette
gespeichert werden, um sie zu  erhalten.
 Sie koennen den Dateinamen    frei waeh
len.                    <Taste>";
520 IF INKEY$="" GOTO 520
530 GOSUB 8010
540 MODE 1
550 PRINT"ENDE":END
1000 REM >eingeben<
1010 CLEAR INPUT:LOCATE #1,1,4:INPUT#1,"
max. 21 Zeichen";x$
1020 LOCATE#1,1,4:PRINT#1,"
                                "
1030 IF LEN(x$)>21 THEN x$=MID$(x$,1,21)
 ELSE FOR i=LEN(x$) TO 20:x$=x$+".":NEXT
 i
1040 IF e=0 GOTO 1140
1050 k=0:mp=10:FOR i=0 TO 9
1060 IF k=1 GOTO 1080
1070 IF r(n(e-1),i)=0 THEN mp=i:k=1
1080 NEXT i
1090 IF mp=10 THEN LOCATE#1,1,4 ELSE GOT
O 1140
```

Programm
DENKNETZ.EXP

```
1100 PRINT#1,"erst loeschen
     <Taste>"
1110 IF INKEY$="" GOTO 1110
1120 LOCATE#1,1,4:PRINT#1,"
                                  "
1130 RETURN
1140 k=0:FOR i=1 TO n
1150 IF x$=w$(i) THEN mn=i:k=1
1160 NEXT i
1170 IF k=0 THEN GOSUB 20010 ELSE GOTO 1
220
1180 w$(ml)=x$:IF e>0 THEN p(e-1)=mp:r(n
(e-1),p(e-1))=ml
1190 IF e=0 THEN n(0)=ml:LOCATE x,1:PRIN
T w$(n(0))
1200 GOSUB 5010
1210 RETURN
1220 IF e=0 THEN n(0)=mn:FOR i=0 TO 9:p(
i)=0:NEXT i:LOCATE x,1:PRINT w$(n(0)):GO
SUB 5010:RETURN
1230 k=0:FOR i=0 TO 9
1240 IF r(n(e-1),i)=mn THEN p(e-1)=i:k=1
1250 NEXT i
1260 IF k=1 THEN GOSUB 5010 ELSE GOTO 12
80
1270 RETURN
1280 p(e-1)=mp:r(n(e-1),p(e-1))=mn
1290 GOSUB 5010
1300 RETURN
2000 REM >loeschen<
2010 CLEAR INPUT
2020 LOCATE#1,1,4:PRINT#1,"
                              "
2030 IF e=0 THEN l=l+1 ELSE GOTO 2120
2040 l(l)=n(0):w$(n(0))="..............
......":LOCATE x,1:PRINT w$(0):FOR i=0 T
O 9:r(n(e),i)=0:NEXT i
2050 FOR i=0 TO n
2060 FOR j=0 TO 9
2070 IF r(i,j)=n(e) THEN r(i,j)=0
2080 NEXT j
2090 NEXT i
2100 n(e)=0
2110 GOSUB 5010:RETURN
2120 FOR i=0 TO 9
2130 IF r(n(e-1),i)=n(e) THEN r(n(e-1),i
)=0
2140 NEXT i
2150 GOSUB 5010:RETURN
3000 REM >blaettern<
3010 IF e=0 GOTO 3100
3020 p(e-1)=p(e-1)+pp
3030 IF p(e-1)>9 THEN p(e-1)=p(e-1)-10
3040 IF p(e-1)<0 THEN p(e-1)=p(e-1)+10
3050 GOSUB 5010
3060 RETURN
3070 FOR w=0 TO 100:NEXT w
3100 n(0)=n(0)+pp
3110 IF n(0)<0 THEN n(0)=n(0)+400
```

Programm
DENKNETZ.EXP

```
3120 IF n(0)>399 THEN n(0)=n(0)-400
3130 LOCATE x,1:PRINT w$(n(0))
3140 GOSUB 5010
3150 RETURN
4000 REM >Ebene waehlen<
4010 LOCATE 18,2*e+1:PRINT" "
4020 e=e+ee
4030 IF e>9 THEN e=e-10
4040 IF e<0 THEN e=e+10
4050 LOCATE 18,2*e+1:PRINT CHR$(246)
4060 FOR w=0 TO 100:NEXT w
4070 RETURN
5000 REM >Ausgabe Begriffskette<
5010 FOR i=1 TO 9:n(i)=r(n(i-1),p(i-1)):
LOCATE x,i*2+1:PRINT w$(n(i)):NEXT i
5020 RETURN
6000 REM >Bildschirmgestaltung<
6010 WINDOW #1,1,40,22,25:PAPER #1,1:PEN
 #1,3:CLS #1
6020 LOCATE x,1:PRINT"..................
..."
6030 LOCATE 1,1:PRINT"Wurzelbegriff"
6040 FOR i=1 TO 9:LOCATE 1,i*2+1:PRINT"A
ssoz-ebene";i:LOCATE x,2*i+1:PRINT".....
...............":NEXT i
6050 LOCATE 18,1:PRINT CHR$(246)
6060 RETURN
7000 REM >Window #1<
7010 CLS#1:PRINT#1,CHR$(240);" ";CHR$(24
1);" Ebene waehlen   ";CHR$(242);" ";CHR
$(243);" blaettern"
7020 PRINT#1,"f1  eingeben          f2  lo
eschen"
7030 PRINT#1,"f7  ausdrucken        f9  En
de"
7040 RETURN
8000 REM >Daten speichern<
8010 CLS#1:PRINT#1,"Benennen Sie die Dat
ei"
8020 CLEAR INPUT:INPUT#1,"mit maximal 8
Zeichen:";n$
8030 IF LEN(n$)>8 GOTO 8010
8040 n$=UPPER$(n$+ex$)
8050 OPENOUT n$
8060 WRITE#9,n,l
8070 FOR i=0 TO l:WRITE#9,l(i):NEXT i
8080 FOR i=0 TO 399
8090 WRITE #9,w$(i)
8100 FOR j=0 TO 9:WRITE#9,r(i,j):NEXT j
8110 NEXT i
8120 CLOSEOUT
8130 RETURN
9000 REM >Daten laden<
9010 ON ERROR GOTO 30010
9020 CLS#1:PRINT#1,"Benennen Sie die Dat
ei"
9030 CLEAR INPUT:INPUT#1,"mit maximal 8
Zeichen:";n$
9040 IF LEN(n$)>8 GOTO 9020
9050 n$=UPPER$(n$+ex$)
```

Programm
DENKNETZ.EXP

```
9060 OPENIN n$
9070 INPUT#9,n,l
9080 FOR i=0 TO l:INPUT#9,l(i):NEXT i
9090 FOR i=0 TO 399
9100 INPUT #9,w$(i)
9110 FOR j=0 TO 9:INPUT#9,r(i,j):NEXT j
9120 NEXT i
9130 CLOSEIN
9140 RETURN
10000 REM >Ausdruck<
10010 CLEAR INPUT:LOCATE#1,1,4:PRINT#1,"
Drucker bereit?                  <Taste>"
:
10020 IF INKEY$="" GOTO 10020
10030 LOCATE#1,1,4:PRINT#1,"
                                        ";

10040 PRINT#8,"Bestand der Datei ";n$
10050 PRINT#8:PRINT#8,"Begriffe mit Refe
renznummern"
10060 PRINT#8,"--------------------------
----------------------------------------
------------":PRINT#8:PRINT#8
10070 FOR i=0 TO n:PRINT#8,USING"###";i;
:PRINT#8,": ";w$(i);:FOR j=0 TO 9:PRINT#
8,USING"#####";r%(i,j);:NEXT j:PRINT#8:N
EXT i
10080 RETURN
20000 REM >freie Wortnummer suchen<
20010 IF l=0 THEN n=n+1:ml=n
20020 IF l>0 THEN ml=l(l):l(l)=0:l=l-1
20030 RETURN
30000 REM >Fehlerroutine<
30010 CLS#1
30020 IF DERR=146 THEN PRINT#1,"Datei ";
n$;" nicht gefunden" ELSE PRINT#1,"Fehle
r"
30030 PRINT#1:PRINT#1,"Zum Fortfahren Ta
ste druecken"
30040 IF INKEY$="" GOTO 30040
30050 RESUME 30
```

Programmbeschreibung
DENKNETZ.EXP

Da das Programm nur mit Ganzzahlen operiert, werden in **Zeile 10** alle Variablen entsprechend definiert. Dadurch werden kürzere Bearbeitungszeiten möglich. Die Variable w$ wird auf 400 (0 bis 399) Elemente dimensioniert. Jedes dieser w$-Elemente kann ein Wort aufnehmen, und jedem vom Benutzer eingegebenen Wort ist dadurch eine Zahl von 0 bis 399 zugeordnet. DENKNETZ kann also maximal 400 Wörter lernen. Jedes dieser Wörter ist als ein »Knoten« des Wörternetzes zu betrachten.
Die Feldvariable r ist doppelt indiziert und besteht aus 400 x 10 Elementen. Der erste Parameter kann eine Zahl von 0 bis 399 sein. Diese Zahlen beziehen sich auf den Index von w$. Der zweite Parameter kann eine Zahl zwi-

Programmbeschreibung
DENKNETZ.EXP

schen 0 und 9 sein. So ist es möglich, für jedes der 400 Wörter zehn Zahlenwerte zu notieren, die Index auf andere Wörter sind. Diese Zahlenwerte sind die Fäden des Netzes, die einen Knoten mit maximal zehn anderen verbinden. Eine Zuordnung r (27,4) = 16 würde bedeuten, daß dem Wort w$ (27) als fünfte (4) von zehn (0 bis 9) Verknüpfungsmöglichkeiten das Wort w$ (16) zugeordnet ist.
Die beim Löschen freiwerdenden Indizes (von w$) werden bei den folgenden Eingaben nicht wieder verwendet würden. Die Feldvariable l merkt sich deshalb die durch das Löschen freigewordenen w$-Nummern. Fehlerfrei kann das Programm theoretisch nur laufen, wenn 400 gelöschte Nummern gemerkt werden können. Eine so groß dimensionierte Variable l würde aber viel Platz im RAM belegen und in der Praxis doch nie voll genutzt. Deshalb wurde l nur für 100 Werte dimensioniert. Würden also hintereinander mehr als 100 Wörter gelöscht, ohne daß zwischendurch eines neu eingegeben wird, würde das Programm nicht mehr fehlerfrei arbeiten.
In **Zeile 20** werden alle w$ mit Pünktchen gefüllt, damit die leeren Anschläge später auf dem Bildschirm zu sehen sind. **Zeile 30** löscht den Bildschirm, wählt eine Farbe und verzweigt in ein Unterprogramm, das die aktuelle Begriffskette auf den Bildschirm schreibt.
Zeile 40 ruft das Unterprogramm auf, in dem die Gestaltung des Bildschirms programmiert ist.
In **Zeile 50** wird das Fenster # 1 gelöscht und mit einer Frage an den Benutzer gefüllt, worauf in **Zeile 70**, bedingt durch die Eingabe des Benutzers, ein Unterprogramm aufgerufen wird, das nach dem Namen der zu ladenden Datei fragt und diese liest. Diese Datei, der das Programm automatisch den Extender ».DND« zuweist, enthält früher eingegebene Wörter (Knoten) mit ihren Referenzzahlen zu anderen Wörtern (Fäden), also das Netz von Wörtern.
Zeile 80 ruft danach ein Unterprogramm auf, das im Fenster # ein Menü ausgibt, aus dem der Benutzer wählen kann. Entsprechend der darauf folgenden Benutzereingabe werden in den **Zeilen 100 bis 170** Unterprogramme aufgerufen. **Zeile 180** schließt mit Rücksprung zu **Zeile 100** eine endlose Schleife.
Durch Aufruf des **Unterprogramms ab Zeile 500** wird das Programm beendet. Dabei wird der Benutzer aufgefordert, einen Dateinamen einzugeben, unter dem die eingegebenen DENKNETZ-Wörter auf die Diskette geschrieben werden. Es ist also theoretisch möglich, beliebig viele DENKNETZ-Dateien anzulegen.

Programmbeschreibung
DENKNETZ.EXP

Das **Unterprogramm ab Zeile 1000** ermöglicht die Eingabe neuer Wörter. Dabei überprüft **Zeile 1030**, ob das eingegebene Wort zu lang ist und kürzt es entsprechend. Füllt es den vorgesehenen Platz nicht, werden die Leerstellen mit Pünktchen ausgefüllt.
Wenn die Eingabe auf der obersten Ebene des Bildschirms, der sogenannten Wurzelebene, erfolgt (e=0), springt **Zeile 1040** zu **Zeile 1140**. Dort wird durch eine FOR-NEXT-Schleife geprüft, ob das eingegebene Wort (x$) schon bekannt ist. Es werden entsprechende Parameterwerte zugeordnet, oder aber es wird das **Unterprogramm ab Zeile 20010** aufgerufen, wo eine noch freie Wortnummer gesucht wird. Nachdem das eingegebene Wort zugeordnet oder neu aufgenommen ist, wird es mit der ganzen Kette der mit ihm verbundenen Wörter ausgegeben.
Wenn das Wort auf einer tieferen Ebene eingegeben werden soll, überprüft das Programm in der FOR-NEXT-Schleife der **Zeilen 1050 bis 1080**, ob dem in der Ebene darüber stehenden Wort überhaupt noch ein Wort zugeordnet werden kann. Jedem Wort können ja nur zehn andere zugeordnet werden. Ist keine weitere Zuordnung mehr möglich, wird dem Benutzer mitgeteilt (**Zeile 1100**), daß er erst durch Löschen Platz schaffen muß. Ist noch Platz für eine Zuordnung vorhanden, erfolgt ein Sprung nach **Zeile 1140**, wo das eingegebene Wort, wie oben erläutert, bearbeitet wird.
Das **Unterprogramm ab Zeile 2000** wird aufgerufen, wenn ein bereits eingegebenes Wort gelöscht werden soll. Dabei werden nicht nur der entsprechende w$ (Knoten) und seine Referenzzahlen auf andere Wörter (Fäden) gelöscht, sondern es müssen auch alle Eintragungen in der Feldvariablen r (Fäden) aller anderen Wörter durchsucht werden, um die Verbindungen dieser Wörter mit dem gelöschten Wort ebenfalls zu löschen.
Das **Unterprogramm ab Zeile 3000** ermöglicht ein Blättern durch die Wörter einer Ebene. Jedem Wort einer höheren Ebene sind bis zu zehn Wörter zugeordnet, aber auf der darunterliegenden Ebene auf dem Bildschirm kann nur eines davon gezeigt werden. Beim Blättern werden diese zehn Wörter nacheinander auf den Bildschirm geholt. Dadurch verändert sich natürlich auch die Ausgabe auf allen noch tiefer liegenden Ebenen.
Das Wählen einer Ebene, auf der dann geblättert, eingegeben oder gelöscht werden kann, erfolgt im **Unterprogramm ab Zeile 4000**.

Programmbeschreibung DENKNETZ.EXP

Nachdem durch Blättern, Eingeben oder Löschen eine Veränderung vorgenommen wurde, muß die dadurch entstandene neue Begriffskette auf den Bildschirm gebracht werden. Das besorgt das **Unterprogramm ab Zeile 5000**.
Das **Unterprogramm ab Zeile 6000** enthält die Gestaltung des Bildschirms, das **Unterprogramm ab Zeile 7000** die Gestaltung des Fensters # 1.
Beim Abspeichern einer bearbeiteten DENKNETZ-Datei im **Unterprogramm ab Zeile 8000** wird der Benutzer aufgefordert, einen Dateinamen einzugeben (**Zeile 8020**). Ist der Dateiname zu lang, wird erneut eine Eingabe erbeten. Sonst wird in **Zeile 8040** der eingegebene Dateiname in Großbuchstaben umgewandelt und der Standardextender (.DND) angehängt. Die **Zeilen 8050 bis 8120** schreiben nun die Variablen w$, r und l auf die Diskette. Das **Unterprogramm** zum Laden einer Datei (**ab Zeile 900**) ist entsprechend aufgebaut, nur daß hier von der Diskette gelesen wird.
Wenn der Benutzer den Menüpunkt »Ausdrucken« wählt, wird das **Unterprogramm ab Zeile 10000** aufgerufen, das alle Begriffe mit ihren Referenznummern zu anderen Begriffen formatiert an den Drucker übergibt (**Zeile 10070**).
Das **Unterprogramm ab Zeile 30000** wird aufgerufen, wenn der Benutzer bei der Funktion »Laden einer Datei« einen Dateinamen angegeben hat, der nicht auf der Diskette zu finden ist (Fehler 146) oder irgendeine andere Fehleingabe erfolgte.

Experten sprechen

Wer gewohnt ist, Probleme nach den Regeln von BASIC zu lösen, wird, mit einer deklarativen Sprache konfrontiert, erst einmal verstummen. Bildlich vorgestellt ist da nicht nur der Klang einer fremden Sprache, den man etwa auf einer Reise in ein europäisches Ausland hören kann, sondern es ist so, als reise ein Europäer nach Asien: Man vernimmt aus den unverständlichen Worten eine fremde Mentalität.

Die Entwicklung der Programmiertechniken: Von der Verarbeitung großer Datenmengen ...

Computer wurden für ganz bestimmte Arbeiten entwickelt. Die Struktur der Programmiersprachen ist eng mit diesen Aufgaben verbunden. In der Anfangszeit der Datenverarbeitung standen die enorm wachsenden Datenmengen im Vordergrund. Eine tiefere Sicht der Forschung, komplexere Methoden der Produktion und ei-

ne dichtere Vernetzung der Gesellschaft hatten sie hervorgebracht.
In den folgenden Jahren verlagerte sich die Software-Entwicklung in Bereiche, in denen der Anteil nichtnumerischer und strukturorientierter Operationen zunahm. Beispiele hierfür sind Textverarbeitungs- und Datenbankprogramme, bei denen die sogenannte »Benutzeroberfläche« durch Anpassung an die natürliche Sprache immer anwenderfreundlicher wurde.
An diesem Punkt der kommerziellen Entwicklung wurden Forschungsergebnisse aus dem militärischen Bereich interessant, wo man sich schon längere Zeit mit dem auseinandersetzte, was heute unter dem Begriff künstliche Intelligenz verstanden wird.
Je mehr sich die Funktionsweisen eines Programms dem alltäglichen menschlichen Leben anpassen sollten, desto mehr entpuppten sich die prozeduralen Programmiersprachen als Hindernis. Deshalb hatte man schon in den 60er Jahren intensiv begonnen, deklarative Sprachen zu erforschen und zu entwickeln.

...zur Selbststrukturierung des Programms

Als eine der erfolgreichsten kann man heute *LISP* nennen. Bei diesem Namen handelt es sich um eine Verkürzung von *List Processing Language* (Listenverarbeitungssprache). Schon hier wird deutlich, daß die Konzeption von LISP auf die Verarbeitung von Strukturen, Listen eben, abgestimmt ist. Der Programmierer wird von der mühsamen Aufgabe entbunden, eine Struktur zu entwickeln. LISP und andere KI-Sprachen laufen natürlich nicht auf einem Heimcomputer. Aber für PC-Besitzer sind sie durchaus schon zugänglich.
LISP scheint unterdessen von seinem Konkurrenten *Prolog* überflügelt worden zu sein. Die Zusammenziehung von *Programming in Logic*, Programmieren in Logik, weist auf den ausgeprägt deklarativen Charakter dieser Programmiersprache hin. Allerdings gibt es inzwischen schon so viele verschiedene Prolog-Entwicklungen, daß die Orientierung schwer fällt.

Mit Turbo Prolog wird eine leistungsfähige KI-Programmiersprache breiten Anwenderkreisen zugänglich

Die Erschließung eines breiteren Marktes wurde durch die Veröffentlichung von *Turbo Prolog* eingeleitet. Es stammt aus dem Hause Borland, das schon mit *Turbo Pascal* Einfluß auf die Software-Entwicklung genommen hat. Die Bedeutung von Turbo Prolog liegt nicht nur in einer Benutzerfreundlichkeit, die alle bislang vorgestellten Prolog-Versionen bei weitem übertrifft. Als vorrangig ist auch nicht die Vielfalt von »Prädikaten« zu nennen, wie die »Befehle« hier heißen. Sie erlauben die einfache Steuerung numerischer Prozesse, des Bildschirms oder des Soundgenerators. Entscheidend ist der Preis, der deutlich unter DM 500,— liegt.

Turbo Prolog gibt es vorerst nur für IBM-kompatible Rechner mit MS- oder PC-DOS ab Version 2.0. Das Programm wird vollständig in den Hauptspeicher geladen und erfordert deshalb auch 384 KByte RAM. Ein Diskettenlaufwerk genügt. Nach dem Laden kann man die Prolog-Diskette durch eine Datendiskette ersetzen.
Die Bedienung des Programms ist recht komfortabel. Die wichtigsten Funktionen des Editors sind über Menüs zu erreichen, die in den Bildschirm eingeblendet werden. Andere stehen über die Funktionstasten zur Verfügung, und der Rest entspricht weitgehend den Control-Codes, wie sie vom Textverarbeitungsprogramm WordStar her bekannt sind.
Das fertige Programm wird auf Tastendruck compiliert. Das geschieht auch bei längeren Programmen sehr schnell. Findet der Compiler einen Syntaxfehler, wird automatisch in den Editor-Modus zurückgeschaltet, und der Cursor steht direkt hinter der beanstandeten Stelle. Dazu wird eine der 148 Fehlermeldungen ausgegeben.
Das Programm besteht aus mindestens drei Teilen. Der erste Abschnitt beginnt mit dem Aufruf »domains«. Hier werden Vereinbarungen über Datentypen, Listen und Dateien getroffen. Die Anweisung »leute = symbol« setzt fest, daß »leute« dem Datentyp »symbol« und nicht zum Beispiel »integer« oder anderen angehören soll.
Im Abschnitt »predicates« wird die funktionale Struktur aller verwendeten Prädikate bestimmt. Hier kann mit »mann(leute)« beschrieben werden, daß sich das Prädikat »mann« auf »leute« bezieht; »vater(leute,leute)« definiert das Prädikat »vater« als eine Beziehung zwischen »leute« und »leute«.
Schließlich folgt in den »clauses« eine Sammlung von Regeln, nach denen das Programm auf der Suche nach Lösungen Entscheidungen treffen soll. In der Form »mann(johann_georg_mozart).« wird beschrieben, daß »johann_georg_mozart« ein »mann« ist; »vater(georg—johann_mozart,franz_alois_mozart).« bestimmt »georg_johann_mozart« als »vater« von »franz_alois_mozart«.
In den clauses können aber auch Variablen und Bedingungen vorgegeben werden: »elternteil(X,Y) if vater(X,Y) or mutter(X,Y).« liest sich so: »X« ist ein »elternteil« von »Y«, wenn »X« ein »vater« von »Y« oder »X« eine »mutter« von »Y« ist.
Oder: »schwester(X,Y) :— frau(X) , mutter(Z,X) , mutter(Z,Y) , X<>Y.« besagt, daß »X« eine »schwester« von »Y« ist, wenn »X« eine »frau« ist und »Z« eine »mutter« von »Y« und »Z« eine »mutter« von »X« und »X« ungleich »Y«.

Im vorhergehenden Unterkapitel wurde das Programm DENKNETZ am Beispiel einer praktischen Anwendung vorgeführt. In Turbo Prolog sieht eine Lösung dieses Problems so aus:

Prolog-Programm Stammbaum

```
/*     >>> Stammbaum Mozart-Weber <<<   */
/*      programmiert in Turbo-Prolog    */
/*        von Brainware, Wiesbaden      */

domains
   leute = symbol

predicates
   mann(leute)
   frau(leute)
   ehepart(leute,leute)
   vater(leute,leute)
   mutter(leute,leute)
   elternteil(leute,leute)
   bruder(leute,leute)
   schwester(leute,leute)
   geschwister(leute,leute)
   grossvater(leute,leute)
   grossmutter(leute,leute)
   grosseltern(leute,leute)
   onkel(leute,leute)
   tante(leute,leute)
   grossonkel(leute,leute)
   grosstante(leute,leute)
   schwager(leute,leute)
   schwaegerin(leute,leute)

clauses
   mann(johann_georg_mozart).
   mann(franz_alois_mozart).
   mann(leopold_mozart).
   mann(wolfgang_amadeus_mozart).
   mann(hofrat_joh_bapt_freiherr
               _berchtold_v_sonnenburg).
   mann(alois_pantaleon_mozart).
   mann(karl_thomas_mozart).
   mann(franz_xaver_wolfgang_mozart).
   mann(fridolin_weber_sen).
   mann(fridolin_weber_jun).
   mann(franz_anton_weber).
   mann(carl_maria_v_weber).
   mann(jos_lange).
   mann(j_haibel).
   mann(franz_hofer).
   mann(fr_s_mayer).
   mann(max_maria_v_weber).

   frau(anna_maria_sulzer).
   frau(maria_viktoria_eschenbach).
   frau(anna_maria_pertl).
   frau(maria_anna_thekla_mozart).
   frau(maria_anna_mozart).
   frau(konstanze_weber).
```

Prolog-Programm
Stammbaum

```
frau(jeanette_mozart).
frau(maria_babette_mozart).
frau(marianne_eva_chellar).
frau(maria_c_stamm).
frau(genoveva_brenner).
frau(aloisa_weber).
frau(sophie_weber).
frau(josepha_weber).
frau(caroline_brandt).

vater(johann_georg_mozart,
    franz_alois_mozart).
vater(johann_georg_mozart,
    leopold_mozart).
vater(franz_alois_mozart,
    maria_anna_thekla_mozart).
vater(leopold_mozart,
    maria_anna_mozart).
vater(leopold_mozart,
    wolfgang_amadeus_mozart).
vater(wolfgang_amadeus_mozart,
    karl_thomas_mozart).
vater(wolfgang_amadeus_mozart,
    franz_xaver_mozart).
vater(fridolin_weber_sen,
    fridolin_weber_jun).
vater(fridolin_weber_sen,
    franz_anton_weber).
vater(fridolin_weber_jun,
    konstanze_weber).
vater(fridolin_weber_jun,
    aloisia_weber).
vater(fridolin_weber_jun,
    sophie_weber).
vater(fridolin_weber_jun,
    josepha_weber).
vater(franz_anton_weber,
    carl_maria_v_weber).
vater(carl_maria_v_weber,
    max_maria_v_weber).

mutter(anna_maria_sulzer,
    franz_alois_mozart).
mutter(anna_maria_sulzer,
    leopold_mozart).
mutter(maria_viktoria_eschenbach,
    maria_anna_thekla_mozart).
mutter(anna_maria_pertl,
    maria_anna_mozart).
mutter(anna_maria_pertl,
    wolfgang_amadeus_mozart).
mutter(konstanze_weber,
    karl_thomas_mozart).
mutter(konstanze_weber,
    franz_xaver_mozart).
mutter(marianne_eva_chellar,
    fridolin_weber_jun).
mutter(marianne_eva_chellar,
    franz_anton_weber).
```

Prolog-Programm
Stammbaum

```
mutter(maria_c_stamm,
   konstanze_weber).
mutter(maria_c_stamm,aloisia_weber).
mutter(maria_c_stamm,sophie_weber).
mutter(maria_c_stamm,josepha_weber).
mutter(genoveva_brenner,
   carl_maria_v_weber).
mutter(caroline_brandt,
   max_maria_v_weber).

elternteil(X,Y) :-
   vater(X,Y) ; mutter(X,Y).

ehepart(X,Y) :-
   elternteil(X,Z) ,
   elternteil(Y,Z) , X<>Y , !.

bruder(X,Y) :-
   mann(X) , mutter(Z,X) ,
   mutter(Z,Y) , X<>Y.

schwester(X,Y) :-
   frau(X) , mutter(Z,X) ,
   mutter(Z,Y) , X<>Y.

geschwister(X,Y) :-
   mutter(Z,X) , mutter(Z,Y) , X<>Y.

grossvater(X,Y) :-
   mann(X) , vater(X,Z) ,
   elternteil(Z,Y).

grossmutter(X,Y) :-
   frau(X) , mutter(X,Z) ,
   elternteil(Z,Y).

onkel(X,Y) :-
   elternteil(Z,Y) , bruder(X,Z).

tante(X,Y) :-
   elternteil(Z,Y) , schwester(X,Z).

grosseltern(X,Y) :-
   grossvater(X,Y) ;
   grossmutter(X,Y).

grossonkel(X,Y) :-
   bruder(X,Z) , grosseltern(Z,Y).

grosstante(X,Y) :-
   schwester(X,Z) , grosseltern(Z,Y).

schwager(X,Y) :-
   mann(X) , geschwister(Y,Z) ,
   ehepart(X,Z).

schwaegerin(X,Y) :-
   frau(X) , geschwister(Y,Z) ,
   ehepart(X,Z).
```

Programmbeschreibung
Prolog-Programm
Stammbaum

Nach dem Start dieses Programms ist zunächst die Aufgabe zu definieren, die gelöst werden soll. Dies geschieht bei Turbo Prolog in einem Programmabschnitt »goals«. Nach dem Start meldet sich das Programm mit einer entsprechenden Frage: »goals:«. Jetzt kann der Benutzer seine Anfrage eingeben, die bei diesem nicht ausgeschmückten Programm in Prädikatform geschrieben werden muß. Die Eingabe lautet: »bruder (wolfgang_amadeus_mozart,X).« Das heißt auf deutsch: Zu welchem »X« steht »wolfgang_amadeus_mozart« in der Beziehung »bruder«? Das Programm antwortet mit:
»X = maria_anna_mozart
1 solutions«
»maria_anna_mozart« ist also eine Lösung für »X«, oder übersetzt ins Hochdeutsche: Wolfgang Amadeus Mozart ist ein Bruder von Maria Anna Mozart, was zweifellos richtig ist. Die Anfrage, das »goal«: »schwester (konstanze_weber,X)« wird mit drei Lösungen beantwortet:
»X = aloisa_weber
X = sophie_weber
X = josepha_weber
3 solutions«
Konstanze Weber hatte drei Geschwister. Das Ziel: »schwager(X,Y)« bringt eine Liste aller Schwager-Beziehungen auf die Mattscheibe. Dabei ist die Bearbeitungszeit so kurz, daß die Antwort ohne spürbare Verzögerung erfolgt. Hat man erst einmal ein Gefühl für die neue Denkstruktur entwickelt und die verfügbaren Vokabeln gelernt, gewinnt man schnell Gefallen an diesem turbo-schnellen Prolog.
Sie können selbst beurteilen, wie einfach diese Struktur im Vergleich zu der des BASIC-Programmes DENKNETZ ist. Dennoch ist das Prolog-Programm in der Lage, nicht nur einen Überblick über den Stammbaum der Familien Mozart und Weber zu geben, sondern auch erschöpfende Auskunft über die Verwandschaftsbeziehungen zu geben. Versuchen Sie sich einmal vorzustellen, wie ein entsprechendes BASIC-Programm aussehen müßte.
Von natürlichen Sprachen behauptet man, daß man sie beherrschen würde, wenn man in ihnen träumt. Bei einer Computersprache sollte man es allerdings nicht soweit kommen lassen, sonst *besteht_gefahr_nicht_sieht(X,schoenheit_natur) and nicht_versteht(leser, autor):—abstraktion > maximum. goal:ist(buch,zuende).*

Literaturverzeichnis

Bundy, A., Praktische Einführung in die Künstliche Intelligenz, Vaterstetten: IWT-Verlag, 1986

Dreyfus, Hubert L., Die Grenzen künstlicher Intelligenz, Königstein: Athenäum, 1985

O'Shea, Tim; Self, John, Lernen und Lehren mit Computern, Basel: Birkhäuser Verlag, 1985

Österle, Hubert, Entwurf betrieblicher Informationssysteme, München: Hanser Verlag, 1981

Radig, B.M.; Dreschler-Fischer, L., Grundlagen und Anwendung der Künstlichen Intelligenz, Hamburg: McGraw-Hill, 1986

Riech, Elaine, Artificial Intelligence, New York: McGraw-Hill, 1983

Savory, S.E. (Hrsg.), Künstliche Intelligenz und Expertensysteme, München: Oldenbourg Verlag, 1985

Schnupp, Peter; Leibrandt, Ute, Expertensysteme — nicht nur für Informatiker, Berlin; Heidelberg; New York; Tokyo: Springer Verlag, 1986

Siekmann, J.H., Einführung in die Künstliche Intelligenz, Berlin; Heidelberg; New York; Tokyo: Springer Verlag, 1982

Stede, M., Einführung in die Künstliche Intelligenz Bd. 1—3, Sprendlingen: Luther Verlag, 1983

Stevens, Lawrence, Auf der Suche nach der künstlichen Intelligenz, Landsberg am Lech: moderne verlagsgesellschaft, 1985

Stoyan, Herbert, Maschinen-unabhängige Code-Erzeugung als semantikerhaltende beweisbare Programmtransformation, Berlin; Heidelberg; New York; Tokyo: Springer Verlag, 1986

Register



Gebrauchsmusterschutz des Druckhaus Achilles, Aachen (DBGM)

Computer verständlich

Intelligenz in BASIC

Sie werden Elektronengehirne und Denkmaschinen genannt, doch können Computer wirklich „intelligent" sein? Der Expertenstreit um diese Frage wird die Gemüter noch lange erhitzen.

Dieses Buch informiert darüber, was unter „Künstlicher Intelligenz" zu verstehen ist. Es gibt Auskunft über Expertensysteme und beschreibt, was diese heute schon leisten können. Anhand zahlreicher Programmbeispiele erfährt der Leser, wie sich Künstliche Intelligenz äußert und wie sie programmiert wird. Alle Beispiele sind in BASIC geschrieben, sorgfältig dokumentiert und leicht nachvollziehbar.

Karl-Heinz Koch ist als freier Publizist tätig. Er schreibt regelmäßig für verschiedene Computerzeitschriften und hat bereits mehrere Computerbücher veröffentlicht.

ISB N 3-8068-4320-1



